A Gap Analysis of an Internet of Things Education Toolkit for Undergraduate Students

A thesis submitted by
Kerrianne Marino

In partial fulfillment of the requirements for the degree of

Master of Science
in
Human Factors
Tufts University

May 2018

Advisor: Chris Rogers

Abstract

National Instruments (NI), a company that produces automated test equipment and virtual instrumentation software, and PTC, a computer and software technology company, partnered with Tufts University to develop an Internet of Things (IoT) platform for undergraduate engineering students. Through interviews with five engineering students, interviews with two developers of the platform, and usability tests with eight other participants in Tufts University's College of Engineering, I use a gap analysis approach to outline the shortcomings in five aspects of the platform that prevented novice users from quickly developing and/or understanding the IoT data systems. These areas include how students approached the getting started process, using the online dashboard, debugging, adding complexity, and sharing data. Errors, mistakes, and misunderstandings from the usability tests largely revolved around the amount of LabVIEW experience the users had. My analysis suggests future iterations of this toolkit so students can develop solutions more successfully and efficiently.

Acknowledgements

I would like to thank National Instruments and PTC for funding and supporting this work. It has been an amazing experience working so closely with such progressive and strong technology companies. I am honored to have been able to contribute to your developing technologies in the education realm.

This entire project would not have been possible without the initiatives of my advisor, Professor Chris Rogers. He has made me remarkably and unexpectedly comfortable in LabVIEW, and has opened my mind to many of the seemingly infinite possibilities of technology, all while helping me grow as a Human Factors student.

Professor Dan Hannon of Tufts University, another member of my committee who particularly strengthened my approaches to Human Factors analyses and gave me constant insight on best practices in this field. Also thank you to Fred Visser of National Instrument, the final member of my thesis committee, who was able to give an extremely helpful industrial perspective of my writings and approaches.

I also need to give a special thank my parents, who have been supportive my whole life, but have also proven to be the best pair of roommates and life coaches I could ask for. Thanks for putting literal and metaphorical fuel in my tank for the past 24 years. To my brother Joey, thank you for understanding my sense of humor and never keeping life boring.

Lastly, I could like to thank the CEEO for always giving me new opportunities to grow as a learner and gave me every resource I needed the past few years. Whether you were giving me live updates on the weather predictions for Marathon Monday, reading over my Literary Review for grammar errors, or simply asking me how I was doing, your support has been a much needed pillar of strength as I completed my final semester of grad school.

Table of Contents

Abstract i

Acknowledgements ii

List of Tables vi

List of Figures vii

Abbreviations ix

Chapter 1 - Introduction 1

1.1 Project Background 1
1.2 What is IoT? 4
1.3 Motives - Industry & Education 5

Chapter 2 - Literary Review 9

2.1 IoT Review 9
2.1.b. On-Demand Storage 12
2.1.c. Presentation 12
2.1.d Security 13
2.2 Other Platforms 14
2.2.a. Novice Users 15
2.2.b. Advanced Users 22
2.3 Gap Analysis Approach 24

Chapter 3 - Purpose 25

3.1 Why Use Gap Analysis 25
3.2 Goals of the Thesis 26

Chapter 4 - Technical Background and Toolkit Logistics 28

4.1 LabVIEW 29
4.2 ThingWorx Cloud 29
4.3 MyRIO 31
4.4 Getting Started Utility 32

Chapter 5 - Methodology and Procedures 41

5.1 Approach 41
5.2 Interviews 46
5.2.a. NI &PTC 46
5.2.b. Undergraduate Robotics Students 47
5.3 Usability Tests 48
5.3.a. Pre-Test 50
5.3.b. Test 50
5.3.c. Post-Test 56

Chapter 6 - Results 63

6.1 Getting Started 63
6.1.a NI/PTC Interviews 64
6.1.b Student interviews 65
6.1.c. User Testing 67
6.2 ThingWorx Dashboard 72
6.2.a Developer Interviews 73
6.2.c. User Testing 77

6.2.d. Summary 78
6.3 Debugging 78
6.3.a. Developer Interviews 81
6.3.b. User Interviews 82
6.3.c. User Testing 82
6.4 Adding Complexity 83
6.4.a Developer Interviews 84
6.4.b. Student Interviews 87
6.4.c. User Testing 88
6.4.d. Summary 90
6.5 Sharing Data 90
6.5.a Developer Interviews 91
6.5.b. Student Interviews 93
6.5.c. Summary 94

Chapter 7 - Analysis 95
7.1 The Getting Started Process 95
7.1.a. Discussion of Findings: Student Interviews 96
7.1.c. Discussion of Findings: Usability Test Participants 102
7.2 ThingWorx Dashboard 131
7.2.a. Developers 132
7.2.b. Users 132
7.2.c. Gaps 133
7.2.d. Summary 134
7.3 Debugging 135
7.3.a. Developer Interviews 135
7.3.b. Student Interviews and Usability Tests 136
7.3.c. Gaps 136
7.4 Adding Complexity 137
7.4.a Developer Interviews 137
7.4.b. Student Interviews 138
7.4.c. Usability Tests 139
7.4.d. Gaps 140
7.4.e. Summary 141
7.5 Sharing data 141
7.5.a Developer and Student Interviews 141
7.5.b. Gaps 142
7.6 Future Conditions on Approaches 142

Chapter 8 - Summary & Conclusion 144

Appendix 149
Appendix A - Usability Test Transcripts 149
Appendix B - Task Assignment Handout 153
Appendix C - Helpful Tips 155
Appendix D - Usability Test Individual Results 158
Appendix E - Interview Full Transcript for Selected Questions 162
Appendix F - Student Interview Questions 165

Bibliography 169

List of Tables

2.1: Summary of IoT Platforms 23

6.1: Student Interview: Getting Started 65

6.2: Usability Test Participants Coding Experiences 68

6.3: Errors in LabVIEW 70

6.4: Student Interview Dashboard Responses 76

6.5: Usability Test Participants Interactions with ThingWorx Dashboard 77

6.6: Student Interview myRIO/ThingWorx Adding Complexity 88

6.7: Usability Test Participant myRIO Opinions 89

D.1: User UP1...158

D.2: User UP2...158

D.3: User UP3...159

D.4: User UP4...159

D.5: User UP5...159

D.6: User UP6...160

D.7: User UP7...160

D.8: User UP8...161

D.9: Total Usability Participant Data..161

E.1: Student Interview Questions and Responses 165

List of Figures

1.1: IoT Education Toolkit Data Flow Diagram....................2

2.1: A simple framework for an IoT system....................10

2.2: Example IoT data flow architecture with multiple pieces of hardware....................11

2.3: IFTTT Applet....................15

2.4: The Particle's Photon Schematic....................16

2.5: Photon Pet Feeder....................17

2.6: MathWork's ThingSpeak Weather Station Example....................18

2.7: Preview of ThingWorx Composer....................20

2.8: LabVIEW IoT Education Toolkit Express VI....................21

4.1: ThingWorx Express VI Window....................30

4.2: MyRIO Specifications....................31

4.3: GSU, Register with ThingWorx....................33

4.4: GSU: ThingWorx Login....................34

4.5: GSU: Connecting Computer & Cloud....................35

4.6: GSU, Adding a Dashboard....................36

4.7: GSU, Connecting to myRIO....................38

4.8: GSU, Checking the myRIO Wi-Fi....................38

4.9: GSU, Connecting myRIO & Cloud....................39

5.1: "Gap" Model of Service Quality....................43

5.2: Marino "Gap" Model of Platform Quality....................44

5.3: Usability Test Participant Response to Question: *How much experience do you have with IoT*....................49

5.5: Usability Test, Task 1 Code....................53

5.6: Usability Test, Task 2 Cod.....................53

5.7: Usability Test, Task 3 Code....................54

5.8: Usability Test, Task 4 Code....................55

5.9: Post-Task Assignment Graphics....................57

5.10: Post-Task Assignment #1....................58

5.11: Post-Task Assignment #2 60

5.12: Post-Task Assignment #3 61

5.13: Post-Task Assignment #4 62

6.1: Usability Test Participant LabVIEW Experience 67

6.2: ThingWorx Dashboard: Values that are sent to or from ThingWorx 73

6.3: LabVIEW Probe 79

6.4: ThingWorx Terminal. 80

6.5: ThingWorx Add Property Privacy Settings 92

7.1: GSU Finding Examples 98

7.2: Opening a VI 105

7.3: Pre-Task Video, Where to Open a VI 106

7.4: LabVIEW Front Panel & Block Diagram 107

7.5: LabVIEW While Loop 108

7.6: LabVIEW Code Running 109

7.7: LabVIEW Broken Wire 109

7.8: MyRIO Button to While Loop Stop Button 110

7.9: LabVIEW Task Bar 111

7.10: IoT Toolkit Express VI Window 113

7.11: No Property Selection: Express VI Window 115

7.12: Property Selection: Express VI Window 116

7.13: No Property Selection: Express VI 116

7.14: Add Property 117

7.15: No Thing Selection: Express VI Window and Resulting Express VI 119

7.16: Add New Thing 120

7.17: IoT Toolkit Menu From Front Panel 121

7.18: IoT Toolkit Thing XControl on Front Panel 122

7.19: Read/Write Toggle on Express VI Window 124

7.20: Read/Write Express VI Toggle 125

7.21: LabVIEW Example Express VI 130

7.22: IoT Toolkit Express VI 130

Abbreviations

AWS	**A**mazon **W**eb **S**ervices
GUI	**G**etting **S**tarted **U**tility
IFTTT	**I**f **T**his **T**hen **T**hat
IoT	**I**nternet **o**f **T**hings
NI	**N**ational **I**nstruments
UI	**U**ser **I**nterface
VI	**V**irtual **I**nstrument

Chapter 1 - Introduction

1.1 Project Background

NI, PTC, TUFTS University Collaboration

National Instruments (NI), a company that produces automated test equipment and virtual instrumentation software (National Instruments, 2018), and PTC, a computer and software technology company (PTC, 2018), took initiatives to partner with Tufts University to develop an IoT platform for undergraduate engineering students to use in the classroom. Professor Chris Rogers from Tufts University, who helped develop the platform, used the tool in his robotics course during the fall of 2017 (National Instruments, 2017). NI's objective was to make it possible for engineering educators to teach IoT concepts using NI academic hardware and software platforms. Using the myRIO Student Embedded Device, LabVIEW development platform, and PTC's ThingWorx cloud environment, students could learn about robotics and IoT in a project-based learning atmosphere. (National Instruments, 2018). They used the toolkit to read and write properties from the myRIO as shown in the figure below (PTC, 2018). Chapter 4 goes into further detail about these devices and software.

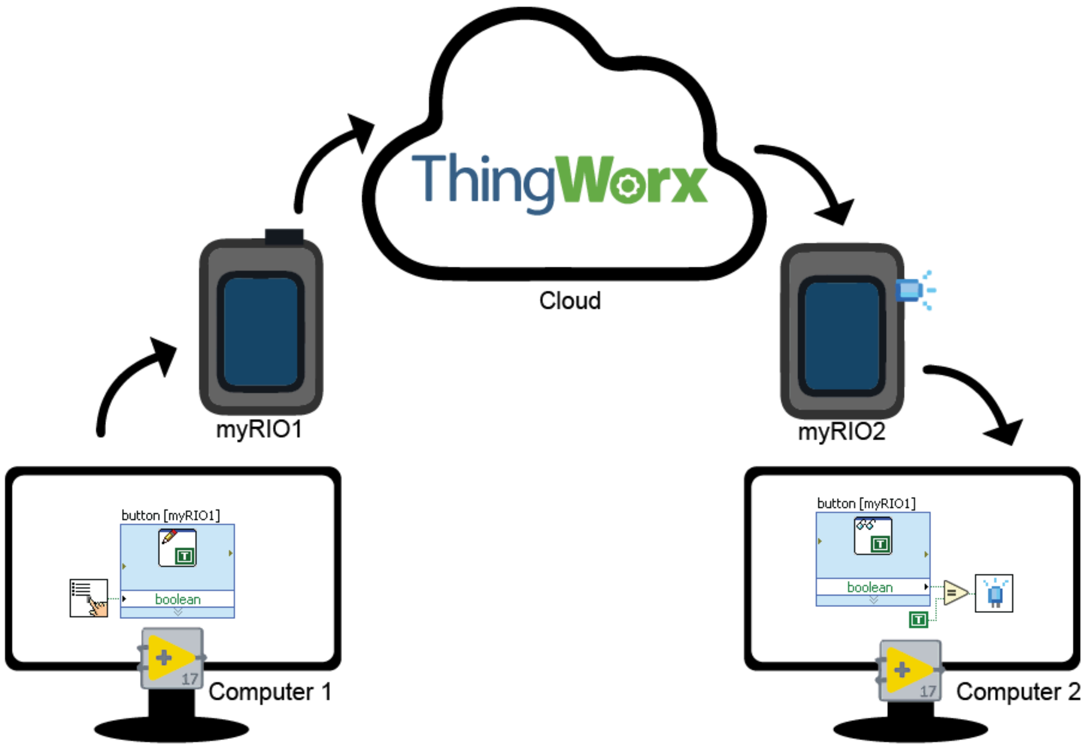


Figure 1.1: IoT Education Toolkit Data Flow Diagram. Example of data flowing from one Thing to another. "myRIO1" is a Thing with property, "button." The user has coded the myRIO on (National Instruments, 2018) so when the button is pressed on the myRIO, the boolean (true/false) information is sent to the cloud. "myRIO2" then reads this button value, and if it reads "true," then the myRIO2's LED will light up"

Figure 1.1 shows an IoT system using this Toolkit. The user would initialize and deploy functions on the myRIO using LabVIEW, and then send (write) or receive (read) information from the cloud that is harnessing Internet data. Here, the user has two Things: myRIO1 and myRIO2. A Thing is an arbitrary label that a user can create for a source of data, and the properties are attributes of that Thing. These two Things are identical in this example, besides their names, their attached hardware (a screen for example), and the code that they are running. To run code from a myRIO, a user must use NI's LabVIEW software. Example code is shown in Figure 1.1 on Computer 1 and 2. MyRIO1 has a property labeled "button," and its true/false value is written to the cloud when someone physically pressed the button on the myRIO1. The myRIO2 is reading this button value, and when it is “true,” a value of “true” is sent to the myRIO2’s LED, turning it on. If the myRIO’s are connected to a Wi-Fi network, a user can run this whole system wirelessly.

Using these elements, this Toolkit allows university students and professors to “build real world IoT applications in the classroom” (National Instruments, 2017). This system allows educators to support and cultivate the influence that IoT is having on education by preparing students to not only immerse in the attitude of the industrial IoT and data acquisition environments, but also help them become effective designers in this world.

The Toolkit’s general features are broken down into five different measurable areas:

1. Getting started process - From opening the platform and setting up a ThingWorx account to connecting the microcontroller and creating measurable data
2. Accessing and using the ThingWorx dashboard, which displays real-time data
3. Debugging the system when it fails to run properly
4. Adding complexity to the student projects by utilizing all the platform’s and myRIO’s technologies
5. Sharing data between users and devices

Gap Analysis

In order to evaluate the effectiveness, efficiency, satisfaction, and utility of the Toolkit, this gap analysis measures the differences between National Instruments’ expectations and delivery of its product, and the users’ needs, expectations, and perceived services as outlined in Polaris Market Research’s “Gap Analysis-the Foundation of Customer Satisfaction Research” (Carlson, 2010). The gap analysis classifies the service and its features relative to user needs and expectations derived via

1. Interviews from the students and developers who have used the system before or helped develop it
2. Usability testing with university engineering students

The analysis section will provide a comprehensive guide to the identified gaps in the section as well as some suggestions on how to approach these gaps to create a more productive system.

1.2 What is IoT?

Smart, connected devices have become an attractive movement in the last few years in almost any market imaginable (Brown, 2016). For example, the Nest Thermostat allows homeowners to adjust the temperature of their house from a mobile app, and with machine-learning algorithms, modifies the thermostat according to the owners' schedules (Nest Labs, 2018). Similarly, Philips Hue Lights lets users change the properties of their lights wirelessly (Philips Lighting Holding B.V, 2018). Both of these devices can work with Amazon's Alexa, a virtual assistant capable of voice interaction, music play-back, providing weather information, setting alarms, and other features (Amazon.com, Inc., 2018). The Nest Thermostat and Philips Hue Lights can serve as actuators for the Alexa, so a user can tell Alexa to "turn off the living room lights" for example, and Alexa will send this command to the Hue Lights. All of these devices are connected to the Internet to allow wireless communication.

Having multiple devices or services interact with one another allows the connected system to optimize its performance. For example, being able to control a home's lights based on the weather, time of day, and the home owner's schedule can save energy and money, and make a more convenient product for the user (she does not have to worry if she remembered to turn the lights off).

With everyday devices becoming increasingly "smarter," a new subject is simultaneously rising to the surface – The Internet of Things. For the purpose of this thesis, I will define the Internet of Things as *the network of smart connected devices, embedded with software, sensors, actuators, and electronics that allow network connectivity to enable these objects to connect and exchange data* (Brown, 2016). The Philips Hue Lights, Nest Thermostat, and Amazon's Alexa are all examples of IoT devices working within a larger IoT ecosystem. These "Things" are Internet-enabled and are intuitive enough to use for a broad range of users (children to older adults in the case of Amazon's Alexa, for example).

1.3 Motives - Industry & Education

Industries are embedding IoT within their operations systems since, when linked properly, the application can maximize overall product and service performance, as outlined in *Harvard Business Review's* "How Smart, Connected Products Are Transforming Competition" (Porter, 2014). For example, John Deere has integrated weather data with crop prices and soil nutrient facts to create an ideal strategy to plant, water, and care for crops. Managing a complex system like this requires training and experience in not only the software and hardware involved, but also the product clouds that are handing all of the integrated data and creating smart logic to control actuators properly.

Education systems preparing students to enter this IoT workplace are beginning to consider these relevant industry needs. As Professor Simon Peyton-Jones, a computer science researcher at the University of Glasgow proposes, we are currently living in not only a biological and physical, but also a digitally connected world (Callaghan, 2012). Teaching students about IoT helps them as young professionals since college students especially are expected to have access to the Internet to complete projects and class assignments. Whether they want to work in an IoT field or not, the world around them is becoming increasingly more “connected,” and at a minimum, they should be aware of the opportunities these big data analytics can create, as well as the safety concerns that can arise (people hacking into accounts with sensitive information, for example).

Furthermore, IoT and the idea of accessing Internet-based data opens up a wide breadth of engineering solutions as it can allow problem-solvers to access data that sensors and physical inputs are pushing to a cloud source. The “cloud” (which can represent many clouds reaching data from their specific sources) holds all the data and manages the logic that can create “smart” commands to actuators. For example, in the case of John Deere’s agriculture system, it includes weather and soil nutrient information (“sensors”) to help dictate seed and crop care with actuators like irrigation systems and tractor control. This system also considers the crop prices in their algorithms to help gauge the amount of demand of each type of crop. Accessing these data further helps reduce waste, and creates an optimal farming environment.

In 2011, Cisco, a leader in IoT, projected that 50 billion objects will be connected to the cloud by 2020 (Evan, 2011). As university engineering students move from their classrooms to their first careers as professional engineers, many of them will be expected to innovate and perform in this IoT environment. In a series of Harvard Business Review articles, PTC's CEO Jim Heppelmann and Michael Porter argue that engineering departments in industry "must add talent in software development, systems engineering, product clouds, big data analytics, and other areas" to meet the high demand of converting and maintaining themselves as a "connected" company (Porter, How Smart, Connected Products Are Transforming Companies, 2014). Therefore, it is particularly important for universities to contribute to the IoT world by teaching their students some form of the subject (usability, technology, data acquisition, etc.) in order to prepare students for this work and life environment (Callaghan, 2012).

Engulfed in a consumer market pushing for all everyday devices to be connected, it is no surprise that one study in 2016 found that 92% of its Bachelor of Science design students had "at least an interest in IoT" (Callaghan, 2012). With personal motivation and relevant workplace environments, students in the STEAM (science, technology, engineering, arts, and mathematics) field have strong motivations to pursue IoT in their undergraduate careers, and industries should consider this audience as an important consumer.

The application of IoT in industry and everyday products has a vast economic benefit, as it allows for optimal experiences and systems, but cloud-based systems are demanding to manage. University students exposed to this IoT ecosystem can prepare themselves for this side of industry. This Toolkit logistically allows students to create simple IoT solutions, and it also strengthens their knowledge and awareness of this IoT space by presenting a platform that can engage sensors or actuators from multiple sources. With the appropriate direction, the Toolkit can guide students in advancing their robotics knowledge by creating “smart, connected” projects.

Chapter 2 - Literary Review

There are many easily accessible sources online for users of all technical levels to create their own IoT solutions. Most of these products focus on the industry or hobbyist market, but the NI/PTC IoT Education Toolkit focuses on the education space. No matter the user, all of these services must contain the proper components for a successfully working IoT system. The IoT Education Toolkit uses NI's LabVIEW and the ThingWorx dashboard for presenting data and giving users a platform to initiate and manipulate these values, the myRIO as its primary source of hardware to process the code, and the ThingWorx Cloud for on-demand data storage where Things easily can read or write values from or to.

2.1 IoT Review

Summarizing *Internet of Things (IoT): a vision, architectural elements, and future directions* (Gubbia J, 2013), to allow clean ubiquitous computing, there are three basic IoT elements involved in any end-to-end solution in which all data streaming and actuation can be done on one application or platform:

1. Hardware
2. On-demand storage
3. Presentation
4. Security

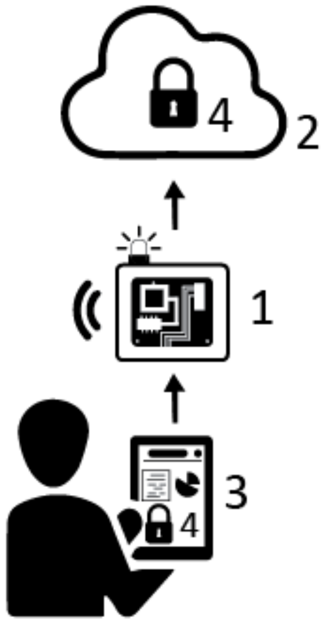


Figure 2.1: A simple framework for an IoT system. To allow users to create their own solutions, IoT services need (1) middleware to process monitored data to send to (2) a cloud storage source by means of (3) an interface or coding platform where the user can control or view the monitored data. These systems should also consider (4) safety in the system to prevent hackers and other users from accessing sensitive information.

2.1.a. Hardware

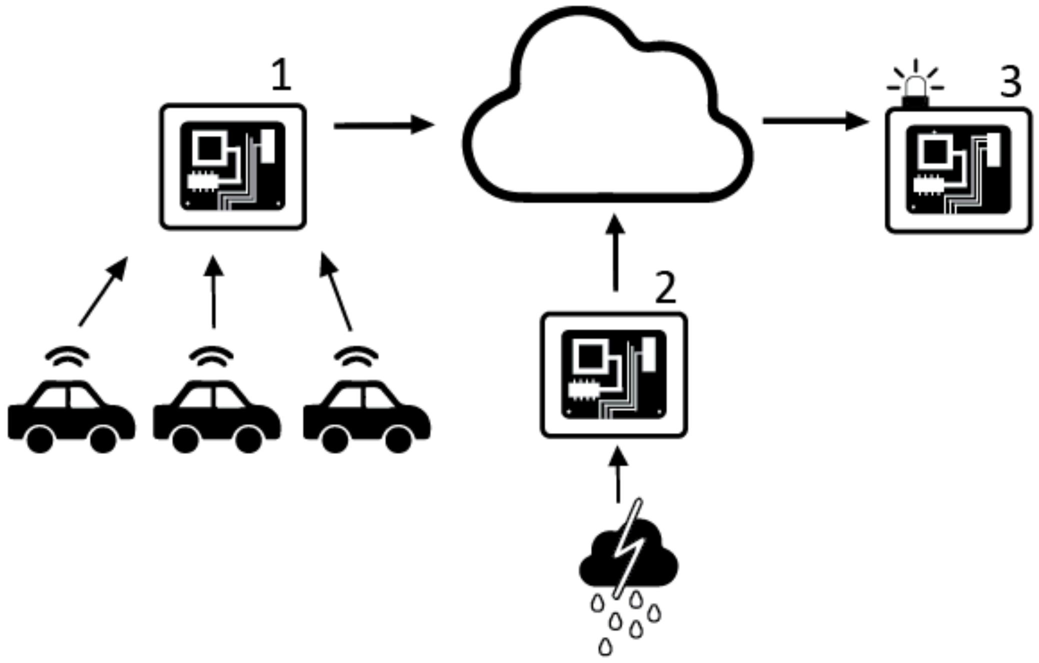


Figure 2.2: Example IoT data flow architecture with multiple pieces of hardware. Two pieces of hardware (1 and 2) sending data to the Internet cloud, which then processes some logic to manipulate the light (activator) on hardware 3.

Hardware for IoT systems includes a combination of sensors, activators, appropriate microcontrollers, etc. that can capture and send or receive Internet data. Values captured from the Internet (i.e. the "cloud") can also stem from physical properties like precipitation in a city and the amount of drivers on the road on a main street. Both of these cases, if part of an IoT system, communicate data through some piece of hardware that can connect to the Internet to communicate with other pieces of hardware, as displayed in Figure 2.2 above. Here, based on specified feedback, the two pieces of hardware are communicating with each other in order to activate the light on the hardware labeled 1.

2.1.b. On-Demand Storage

Middleware, where all data can flow through from multiple nodes, is another element to an IoT platform typically necessary for a complete end-to-end solution. This includes on-demand storage that contains computing tools for data analytics. Figures 2.1 and 2.2 show a cloud to represent this space. It can collect data from multiple sources and provide feedback to the user. Middleware can also incorporate machine learning and complex algorithms for automated decision-making.

2.1.c. Presentation

The data also need a form of visualization to allow the user to easily interact with the monitored environment. This (often times interactive) display can come in the form of an online dashboard. Figure 2.1 shows the user manipulating a data display. Dashboards usually allow users to customize this presentation of data to their liking, as well as, when appropriate, allow users to stimulate an actuator to either ensure successful connections or trigger commands. For example, the user in Figure 2.1 can turn on the light himself to make sure there is an appropriate connection between him and the hardware.

A presentation can also involve the coding platform that allows users to control the logic of their hardware. While a dashboard is typically more intuitive, coding platforms grant users power to create more complicated algorithms and data structures. For example, the user represented in Figure 2.1 can change the piece of hardware by adding an input – a photoresistor that can read a brightness value. He can then change the code to initialize this new sensor. While dashboards can vary in complexity depending on the user and the platform, more complex coding programs can help the user create more intricate IoT projects.

2.1.d Security

Security and reliability are other common themes and priorities for IoT solutions and architectures. If a system undergoes a node failure, it may be essential that it fix itself for safety reasons. For example, if there is an error in an automated assembly line, the system has to realize there is an object out of place and adjust it so the error does not carry over to other part of the route. Furthermore, having sensitive or personal data in the cloud that is protected from hackers (malicious or not) is crucial for a safe and reliable platform.

These securities settings can also involve user login credentials that give access to specific parts of the IoT system (shown in Figure 2.1 on the user's tablet), which can create unwanted barriers for users trying to maneuver around the platform quickly. In education platforms, developers sometimes hold security with less priority than typical systems since project-based courses typically encourage open collaboration, and students are not to share private information with each other.

IoT operations can extend to power plants and traffic lights, which are systems that, if tampered with, can cause immediate and serious harm to people. Students in a classroom are not creating projects that could create imminent harm, but having security settings embedded in an education IoT product can instill this best practice for students. Establishing a habit of working within a secure system is appropriate even if it is unnecessary since the goal of an IoT education toolkit is to expose students to some of the real-world solutions of IoT.

2.2 Other Platforms

This section outlines systems that allow users to connect their devices to a cloud network and create end-to-end solutions (those that enable a user to develop an IoT system with its own hardware and software machinery). There are hundred of IoT development kits and platforms for users of varying levels of expertise, but this sections highlights some examples of the most common ones.

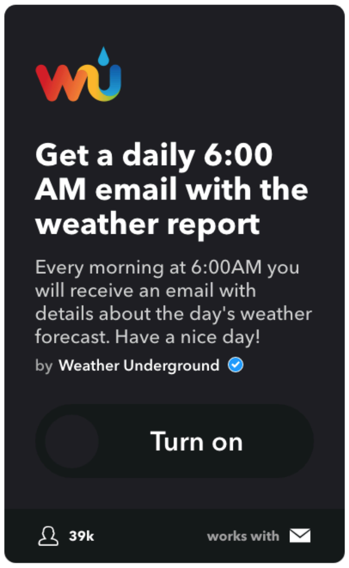
The platforms described in the “Novice Users” section target audiences that may have no experience in IoT or coding, but allow users to create appropriate end-to-end IoT projects that fit a varying level of expertise. These products are typically aimed toward students or hobbyists and are often times open source, meaning that people can use, distribute, and modify the code and schematics without a price.

The target populations for the platforms outlined in the “Advanced Users” section are for consumers more comfortable and trained on IoT platforms, allowing them to develop more advanced data analytics and complex IoT architectures involving machine-learning algorithms for example.

2.2.a. Novice Users

The following platforms and services are aimed for users who could range from no IoT or coding experience to users a bit more practiced in these areas, but not professionals. Having familiarity with these platforms and development kits allows students and hobbyists to gain elementary exposure to IoT systems, which they can eventually build upon with their growing experience.

IFTTT



Get a daily 6:00 AM email with the weather report

Every morning at 6:00AM you will receive an email with details about the day's weather forecast. Have a nice day!

by Weather Underground

Figure 2.3: IFTTT Applet: Sends users an email at 6:00am daily with the weather report (IFTTT, 2018)

If This Then That (IFTTT) is a web-based service to create simple conditional statements, called applets. An applet can easily connect different web services to personal devices

(IFTTT, 2018). For example, it can allow users to receive an email every morning with the weather report (Figure 2.2). IFTTT is a simple platform with an easy to use interface, and no coding or IoT experience is required for successful use. This service gives people of all interests and backgrounds exposure to the IoT world, giving them opportunities to connect practical or entertaining information with their everyday devices and services.

Photon

Particle, an IoT company that develops simple hardware for its integrated IoT platform, is another option that calls for little to no coding experience to develop an end-to-end IoT solution (Particle, 2018).

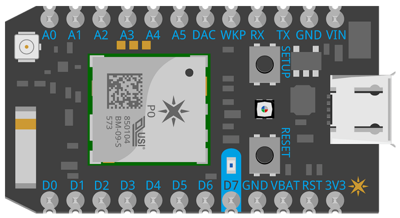

A0 A1 A2 A3 A4 A5 DAC WKP RX TX GND VIN
SETUP
RESET
D0 D1 D2 D3 D4 D5 D6 D7 GND VBAT RST 3V3

Figure 2.4: The Particle's Photon Schematic (Particle, 2018). Small micro-controller with Wi-Fi connection suitable for novice IoT developers

Particle offers an open-source online development environment, where users can code their devices all through an Internet webpage using a texted-based programming language called Wiring. The Photon (Figure 2.3 shown above) is one of Particle's pieces of hardware that is only a few inches long, has built-in Wi-Fi, and pairs well with the Particle coding and IoT environment. Users can develop basic circuit systems to make their projects more dynamic, and manipulate these inputs and outputs via an online coding platform.

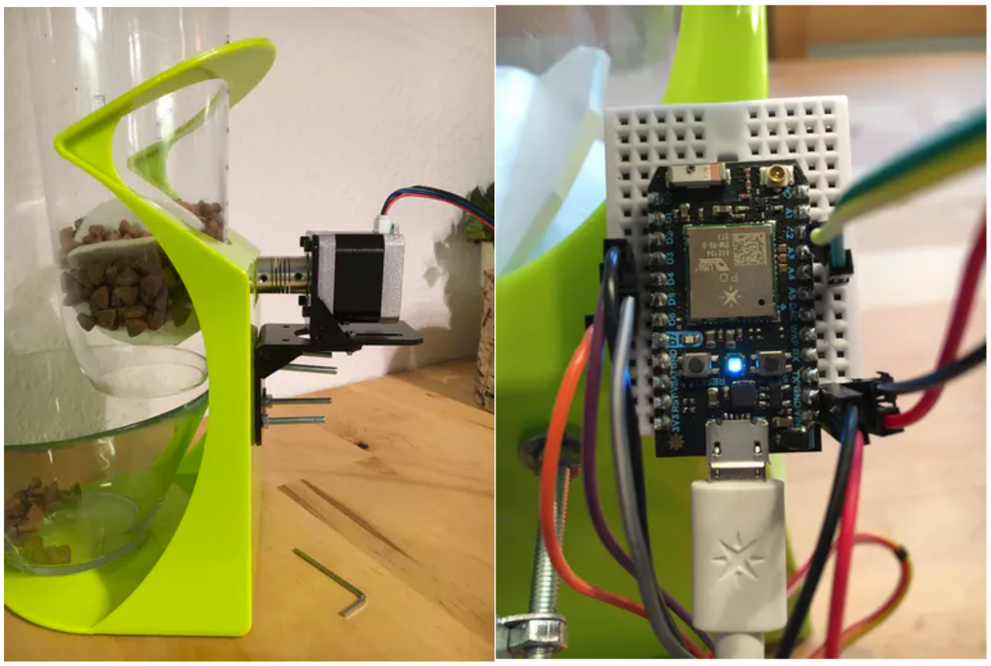


Figure 2.5: Photon Pet Feeder: Example of a Particle Photon project, a remote pet feeder (left) controlled by a servomotor (right) (Lohs, 2017).

Particle products like the Photon are appropriate for novice users in IoT, coding, robotics, and circuitry. They are also especially useful for hobbyist creating simple and inexpensive projects, like a small remote pet feeder that a user can control from a mobile application (Figure 2.5) (Lohs, 2017).

ThingSpeak

MathWorks, a mathematical computing software company, has developed an open-source IoT platform called ThingSpeak (The MathWorks, Inc., 2018). This service allows users to upload and analyze data from the MATLAB computing software, a familiar development environment for many engineering students. Users can store their data in the cloud and visualize and analyze data in MATLAB. This platform pairs well with many commonly used hardware devices like Arduino and Raspberry Pi.

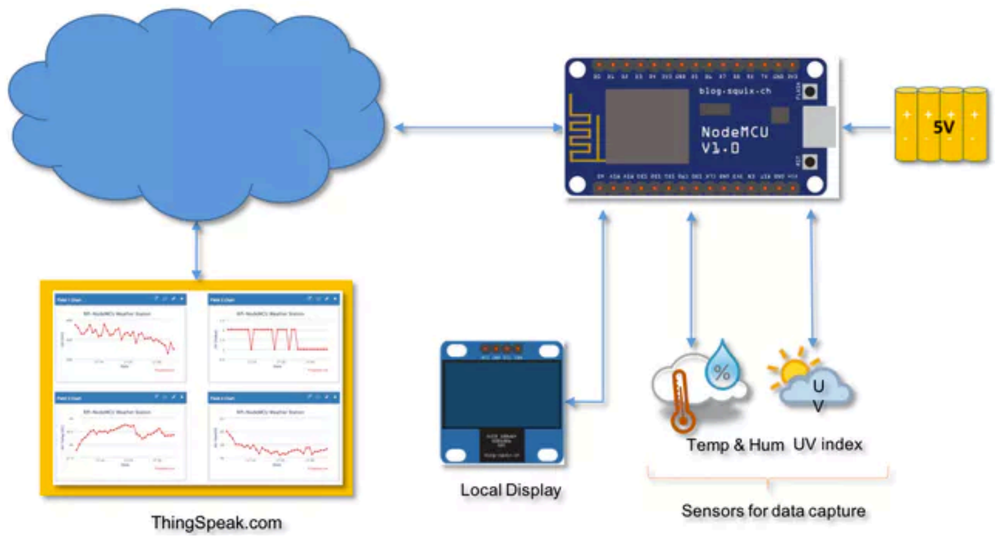


Figure 2.6: MathWork's ThingSpeak Weather Station Example: Data flow and hardware for weather information gathered from a breakout board, and then aggregated and displayed on ThingSpeak (Hackster.io, 2018)

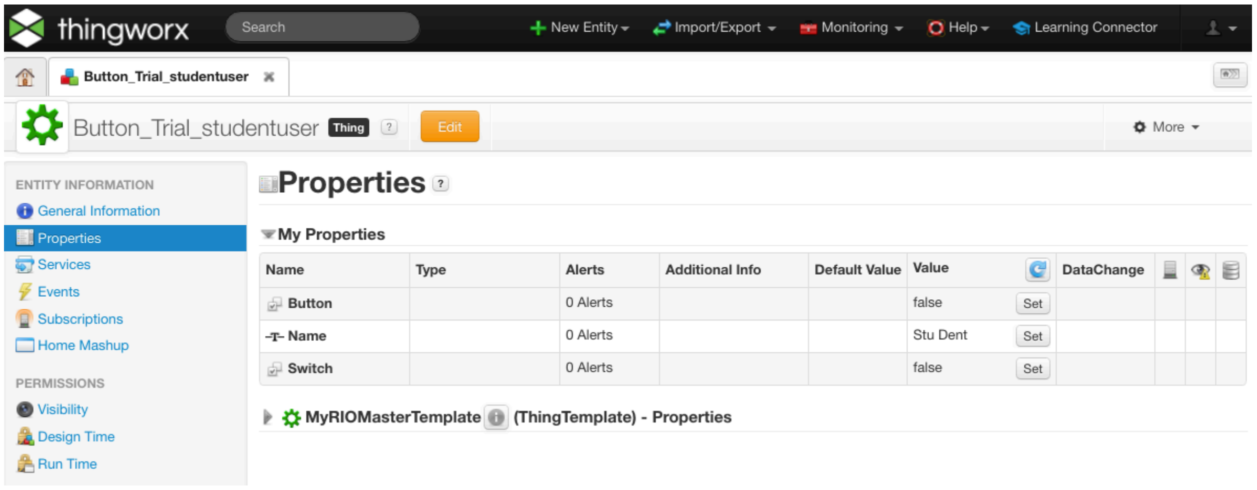
Figure 2.6 above shows an example of a ThingSpeak project. A generic breakout board is wired to sensors capturing different weather information, sending that information through the cloud to ThingSpeak, which then groups the data into intuitive graphs.

This system requires more coding experience to initialize the sensors, and while the Photon and IFTTT is appropriate for users new to coding, ThingSpeak is a product that would be more useful for a slightly more advanced market. MATLAB is one of the top ten programming languages and is offered to students on thousands of university campuses worldwide, making it an accessible and common platform (The MathWorks, Inc., 2018). With the addition of ThingSpeak, this population of students who are already comfortable with MATLAB can easily integrate their projects to IoT by connecting their MATLAB data.

ThingWorx

ThingWorx, PTC's IoT platform, allows companies to source and arrange their data and use data analytics to improve customer experience and optimize business processes by increasing efficiency and improving support and reliability (PTC, 2018). Parts of ThingWorx like the Composer, which can create end-to-end IoT solutions, can be accessed and learned quickly. However, to maximize all of ThingWorx's capabilities in IoT solutions, users would need training and coding experience in JavaScript, especially when trying to access machine-learning and data analytic abilities. Parts of Composer, as described in the *ThingWorx in the Toolkit* section below, is brought down to a user-friendly approach in LabVIEW, so novices in ThingWorx do not have to involve themselves too heavily in the complex architecture of the website.

ThingWorx Composer



Name	Type	Alerts	Additional Info	Default Value	Value		DataChange
Button		0 Alerts			false	Set	
Name		0 Alerts			Stu Dent	Set	
Switch		0 Alerts			false	Set	

Figure 2.7: Preview of ThingWorx Composer: The Thing “Button_Trial” under the “studentuser” account has three properties. “Button,” “Name,” and “Switch,” which users can manage from this page

Composer (Figure 2.7) stands as a development environment for designing and testing IoT applications without relying heavily on coding. It serves as a platform to analyze and control data to and from remote devices, like sensors and actuators from microcontrollers, or more powerful industrial equipment. Users start by creating a “Thing,” which represents an object or source of data that is connected to the cloud. A Thing can have properties, which characterize behaviors or attributes of the Thing. A micro-controller would be an example of a Thing and the sensors and activators associate with it would be its properties. Figure 2.7 shows the Thing “Button_Trial” under the “studentuser” login name. This Thing has the properties “Button,” “Name,” and “Switch.”

Composer also offers more advanced features like linking other data that are reachable from the Internet and adding logic. For example, a user could connect weather data from an open source weather station Internet service that switches his property "Switch" from false to true if there is a chance of rain of 50% or higher. More complex features like this weather logic example are useful for industrial IoT solutions, but on this website may be too complicated for novices to develop.

ThingWorx in the Toolkit

ThingWorx's purpose, as it relates to the Toolkit, is to allow the data that is connected to the Internet to flow through the cloud to communicate with another piece of hardware or software, or to the cloud itself, as shown in Figure 2.1.

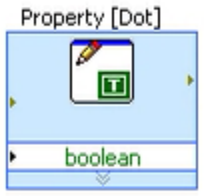


Figure 2.8: LabVIEW IoT Education Toolkit Express VI: Allows users to configure setting interactively through a dialogue box

In theory, users would not need to have any experience with ThingWorx or Composer to easily create a simple IoT project with this Toolkit. It works behind the scenes and presents itself through LabVIEW pieces of code called Express VI's, shown in the figure above. This user interface structure allows students to create new Things and properties, and adjust security settings for viewing/editing data. Users could access the dashboard, ThingWorx Toolkit developers hoped, to monitor data. The dashboard would serve as a visualization center for the data that users choose to send up to the cloud, instead of trying to manipulate data through the more complex architecture in ThingWorx Composer.

Composer (see Figure 2.7) fits as a more advanced and intricate version of the cloud aspects of the Toolkit. PTC Toolkit developers hoped that students would, at some point in their semester, access Composer to create more complicated operations, such as reaching outside data sources and creating logic or a service (a specific function that a Thing can perform). For example, users can access weather data from an open source on the Internet to control some output value. I discuss this approach further in the Results chapter. Users can access online tutorials to learn how to use Composer.

2.2.b. Advanced Users

The following systems are popular platforms for industrial solutions. They feature advanced capabilities for highly trained software engineers to develop smart-connected systems for large companies. Having exposure to the architectures of simpler platforms beforehand, users transitioning to an industrial environment can shift to these more complex systems with less difficulty.

AWS

Amazon Web Service (AWS) is an IoT platform that offers on-demand cloud computing, or the use of a network of servers hosted on the Internet to store, manage, and process data (Amazon Web Services, Inc, 2018). Although the pricing is cost-effective for big businesses, the service is not open-source. Recently, AWS has partnered with common hardware manufacturers like Texas Instruments and Intel to create starter kits compatible with its platform. AWS makes available a vast number of distinct cloud technologies as services, which makes the platform robust, but not necessary or appropriate for novice developers.

Azure & Google Cloud

Other powerful and popular IoT platforms that offer end-to-end solutions are Microsoft's Azure and Google Cloud. Both systems offer similar solutions to AWS in terms of cloud computing, and database connection (Google, 2018) (Microsoft, 2018). Google Cloud also offers big data analytics and machine learning applications to create a more efficient and "smart" IoT system. These systems offer different pricing options, but none are free, and like AWS, are robust and possibly too complex for a novice user.

Summary

Table 2.1: Summary of IoT Platforms

Platform	Novice Users	Coding Experience Needed	Advanced Users (industrial use)	Open Source
IFTTT	Yes	None	No	Yes
Photon	Yes	Some (Wiring)	No	Yes
ThingSpeak	Yes	Some (MATLAB)	No	Yes
ThingWorx Composer	No	JavaScript	Yes	No
ThingWorx Toolkit	Yes	No	No	No
AWS	No	Yes	Yes	No
Azure	No	No	Yes	No

2.3 Gap Analysis Approach

To create an approach for carrying out a gap analysis, “A Gap Analysis Methodology for Product Lifecycle Management Assessment” (Marra, 2018) made a comprehensive generic six-step process among several different approaches from the literature. The article proposes these phases as follows:

1. Planning. Identifying the subject and scope of the analysis, like the system requirements of example.
2. Identify the benchmarking partner. Usually products or services are matched with that of another company in order to maximize performance.
3. Data collecting. Gathering qualitative and quantitative data from academic papers, online databases, questionnaires, and interviews.
4. Determining the gap. Identifying the disparity between the current product and the desired or ideal product.
5. Acting. Implementing the changes that would reduce the gap, or make the product more usable, efficient, enjoyable, or effective.
6. Feedback. After making the changes, developers or marketers should monitor the progress to ensure the results are in line with their expectations and that of the users.

This thesis will implement some of these approaches, as outlined in the Methodology & Procedures Chapter.

Chapter 3 - Purpose

This chapter describes the motives and objectives of this project and the advantages of adopting a gap analysis approach to evaluate the platform. Typically, gap analyses involve benchmarking, a method to compare two companies or products with one another, but I assume a slightly different angle to examine the platform.

3.1 Why Use Gap Analysis

According to Polaris Marketing Research Inc. (Carlson, 2010), a gap analysis can serve as the foundations for customer satisfaction research, which is useful in measuring market positioning relative to major companies. However, even though big software/IoT companies (NI and PTC) funded this project and research, the primary motive of this thesis is to highlight the gaps or shortcomings from an educational perspective. This thesis aims to convey an analysis to benefit general IoT education Toolkit development and delivery rather than competitively compare the product relative to others on the market. For example, a complete gap analysis may consider advertisements and company evaluation, but this investigation excludes these considerations since the users (students) were not given a choice in class. Furthermore, the Analysis Chapter makes suggestions to improve the platform, but this analysis does not take action to implement the changes to the Toolkit, or monitor the feedback and outcomes that a new release would bring. Chapter 7 does however use some approaches and analysis techniques that mirror that of gap analyses motivated by business objectives. This method helps categorize the shortcomings or areas of concern of the toolkit so developers can target the areas they need to improve upon in order for the Toolkit to function at its full potential.

3.2 Goals of the Thesis

The goal of this thesis is to use a gap analysis approach to determine basic needs in an IoT development Toolkit for undergraduate engineering/robotics classrooms, and especially how these needs align with the IoT education platform at hand. Developers released this Toolkit as a beta version in June 2017, but this thesis can give advice on future iterations of this version, as well as serve as a point of reference for other industries interested in creating similar services. The Analysis use the basic outline from "A Gap Analysis Methodology for Product Lifecycle Management Assessment" (Carlson, 2010) mentioned in Section 2.3: Gap Analysis Approach. This report will center around the following modified steps:

1. Planning. I identify the platform and its specifications in section 4.1
2. Identify the basic and specific needs of the Toolkit. This step is included to replace "identifying the benchmarking partner," since this analysis is not aiming to competitively position the Toolkit relative to other platforms from a marketing standpoint.
3. Data Collecting. This step involves student-user interviews from a robotics class that used the platform, interviews with NI and PTC Toolkit developers, usability tests with suitable ideal users, and pre and post questionnaires with these usability test participants are presented in Chapter 5.

4. Determining the gap. The areas that separate the ideal state of the platform (from a developer and user perspective) from the platform’s actual abilities and deliverables are identified in this section. These determinations are in the Results Chapter.

The Analysis and Conclusion (Chapters 7 and 8) make suggestions to improve on the current system, but I do not propose implementing any new features on the Toolkit. Therefore, steps 5 and 6 from “A Gap Analysis Methodology for Product Lifecycle Management Assessment” are excluded.

Chapter 4 - Technical Background and Toolkit Logistics

The developers designed the IoT Toolkit to help university-level engineering students create IoT solutions in a classroom setting. Fulfilling the basic needs of a functional IoT platform, the system contains the following:

1. A coding environment (LabVIEW) to write commands and data logic as well as setup hardware devices.
2. A cloud network (ThingWorx) to receive and send data from/to the coding environment, hardware devices, and/or other actuators.
3. Hardware (the myRIO) acting as a stand-alone device to send and receive commands to/from the cloud to the coding software.

For the beta release, Tufts University students were the target audience, with Dr. Chris Rogers, a mechanical engineering professor and technical developer of the Toolkit, as the instructor. This professor taught the Advanced Robotics course at Tufts University, and the class contained sixteen students. The developers conducted several informal usability tests from January 2017 through August 2017 to discover any initial design approach flaws that could lead to usability errors before the June 2017 beta release. The Toolkit version that the robotics course used contained some iterations and software improvements made throughout the summer until September 2017, when the class began.

4.1 LabVIEW

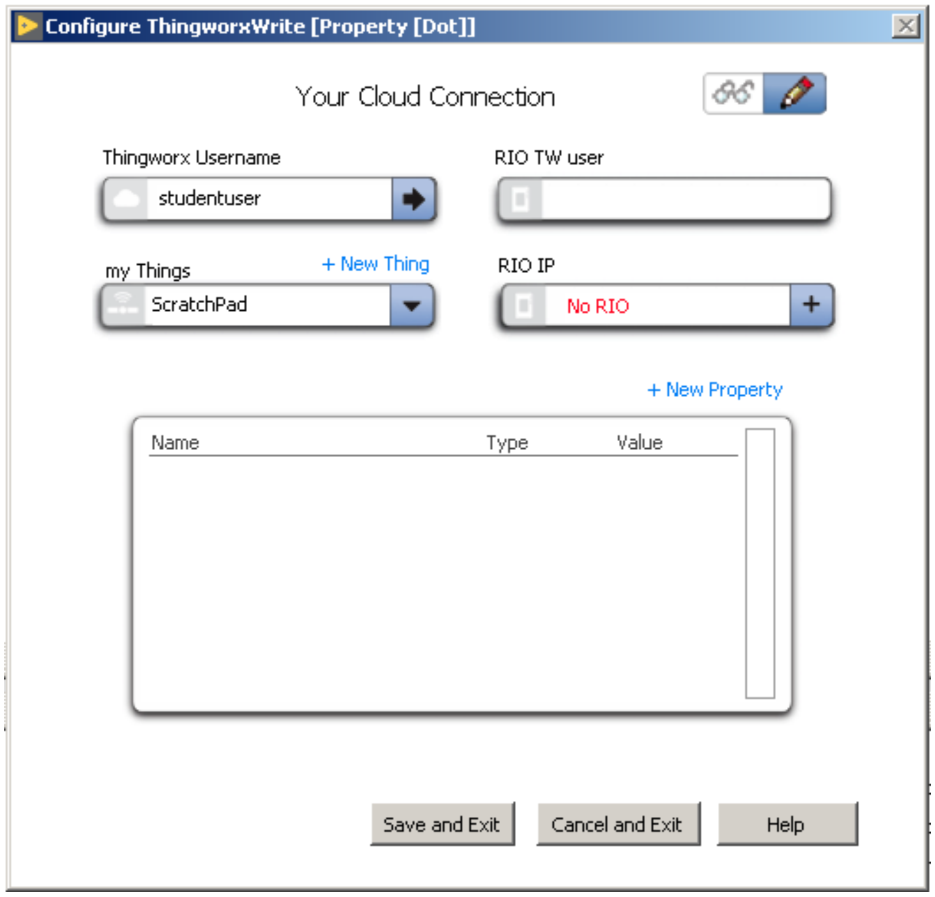
This professor typically uses NI's LabVIEW (Laboratory Virtual Instrument Engineering Workbench) (National Instruments, 2018) in his robotics and engineering classrooms, and every student in the class had access to the software through their Tufts credentials. Because of its availability to this set of users, its ease of use for first time users, and Tufts' contracted partnership with NI, the platform uses this environment as the coding software to work with the Toolkit. LabVIEW features a graphical programming approach to initializing and analyzing data measurements and controls. It uses virtual instruments (VI's) to develop sub-programs for a cleaner and simpler graphical program architecture. A VI can have inputs (controls & constants) and outputs (indicators) to correspond to the piece of code it is referring to/running.

Users can download the software on several different operating systems including Microsoft Windows, and Mac OS. NI offers a low-cost Student Edition of LabVIEW for educational institutions, and features an online community of users and active online forum.

4.2 ThingWorx Cloud

PTC's ThingWorx is the cloud network that the IoT Toolkit utilized in creating its solutions. It allows users to connect devices to the Internet cloud, analyze data, and build and deploy programs. The Toolkit however fails to take advantages of all of

ThingWorx's capabilities for novice users. The Toolkit mostly uses it for connecting devices and their data, and communicating this data to other solutions. ThingWorx is connected through LabVIEW so users can easily connect their data on one coding environment.


Configure ThingworxWrite [Property [Dot]]

Your Cloud Connection

Thingworx Username: studentuser

RIO TW user:

my Things: ScratchPad

RIO IP: No RIO

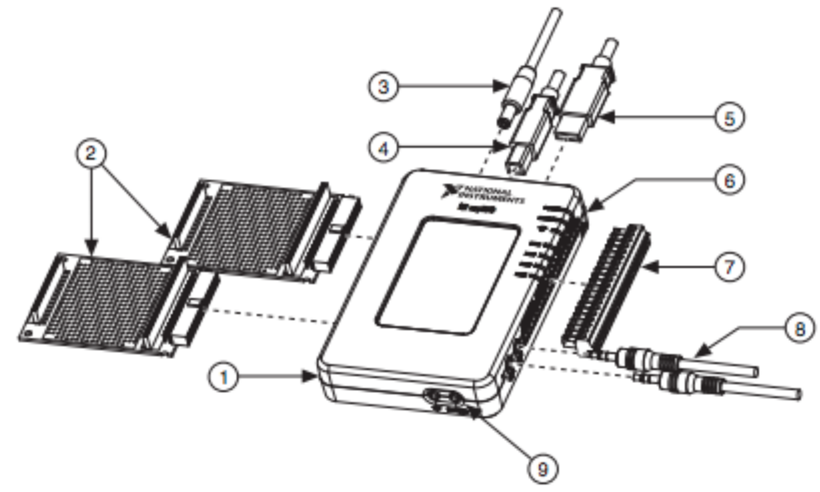
Name	Type	Value

Figure 4.1: ThingWorx Express VI Window: Dialogue box for users to enter and select Things and properties to read or write to or from ThingWorx, as well as set login credentials

To read and write data, ThingWorx has its own set of "Express VI's," where users can configure specific settings for a VI through a dialogue box, requiring minimal coding experience. Figure 3.1 above shows the Express VI window that initializes Things and

Properties to send to the cloud. Here, users can log into their ThingWorx accounts, create or choose Things, initialize their myRIO's, and create or choose new properties.

4.3 MyRIO



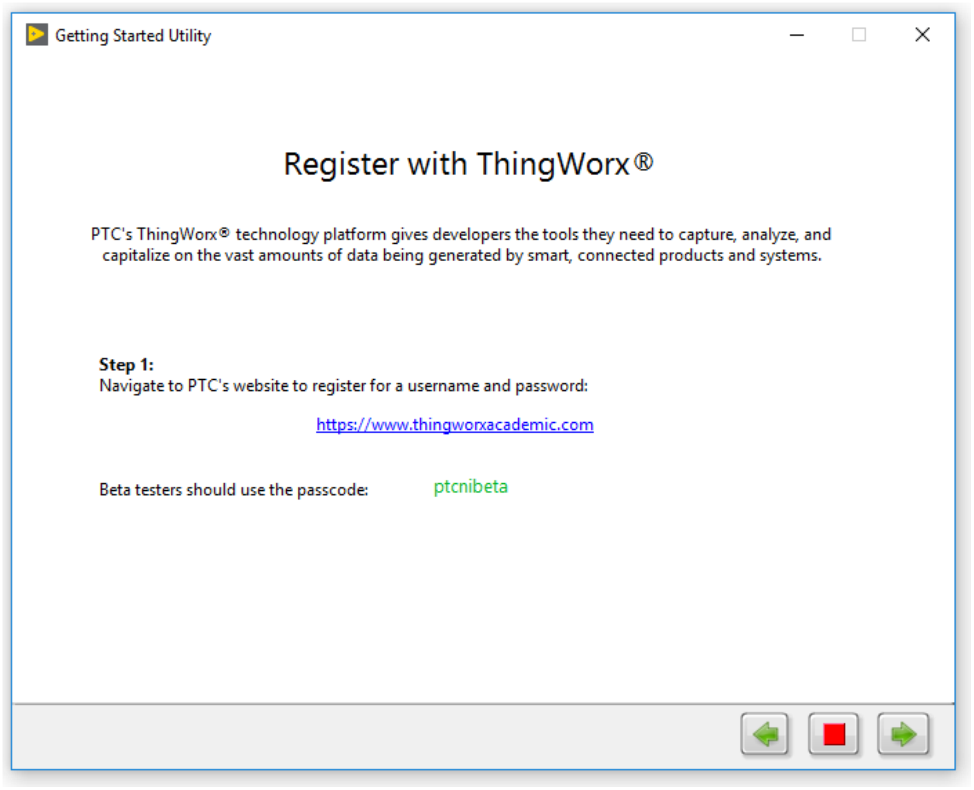
1	NI myRIO-1900	6	LEDs
2	myRIO Expansion Port (MXP) Breakouts (One Included in Kit)	7	Mini System Port (MSP) Screw-Terminal Connector
3	Power Input Cable	8	Audio In/Out Cables (One Included in Kit)
4	USB Device Cable	9	Button0
5	USB Host Cable (Not Included in Kit)		

Figure 4.2: MyRIO Specifications (National Instruments, 2018): National Instruments hardware that pairs with the IoT Education Toolkit. Has Wi-Fi connection and uses LabVIEW as its development platform

The Toolkit's software is paired with the myRIO hardware device to run applications from a stand-alone microprocessor (National Instruments, 2018). The myRIO features analog and digital input/outputs, LED lights, a push button, an onboard accelerometer, Wi-Fi support, and an FPGA module. A FPGA (field-programmable gate array) allows users to optimize complex coding systems by simplifying input/output (I/O) data communication, and is particularly powerful in complicated robotics systems as it can deploy fast commands. The myRIO is easily programmable with LabVIEW and once initialized with ThingWorx configurations, can run programs on its own that communicate to and from the cloud. This method, compared to running programs on a computer or laptop, allows students to create freestanding and mobile IoT projects. Figure 4.2 shows the myRIO specifications.

4.4 Getting Started Utility

The Toolkit features a Getting Started Utility (GSU) in LabVIEW tools, which students must use to create a ThingWorx account, see their data connected to the cloud via a “blink” program, and download the package software on the myRIO. It checks the firmware version on the device, and installs the latest software update if necessary. This Utility also gives the user account an API (application programming interface) key, which acts as an identifier for the ThingWorx platform to distinguish each account. However, users never see this key, but sign into their accounts through their usernames and passwords they create. They build these credentials in the first step of the Getting Started Utility showing below in Figure 4.3.



Getting Started Utility

Register with ThingWorx®

PTC's ThingWorx® technology platform gives developers the tools they need to capture, analyze, and capitalize on the vast amounts of data being generated by smart, connected products and systems.

Step 1:
Navigate to PTC's website to register for a username and password:

https://www.thingworxacademic.com

Beta testers should use the passcode: ptcnibeta

Figure 4.3: GSU, Register with ThingWorx: Overview of ThingWorx and information for users to create an account

Here, users visit the PTC website to create a ThingWorx account, and then they can enter their login credentials on the program in LabVIEW (shown below).

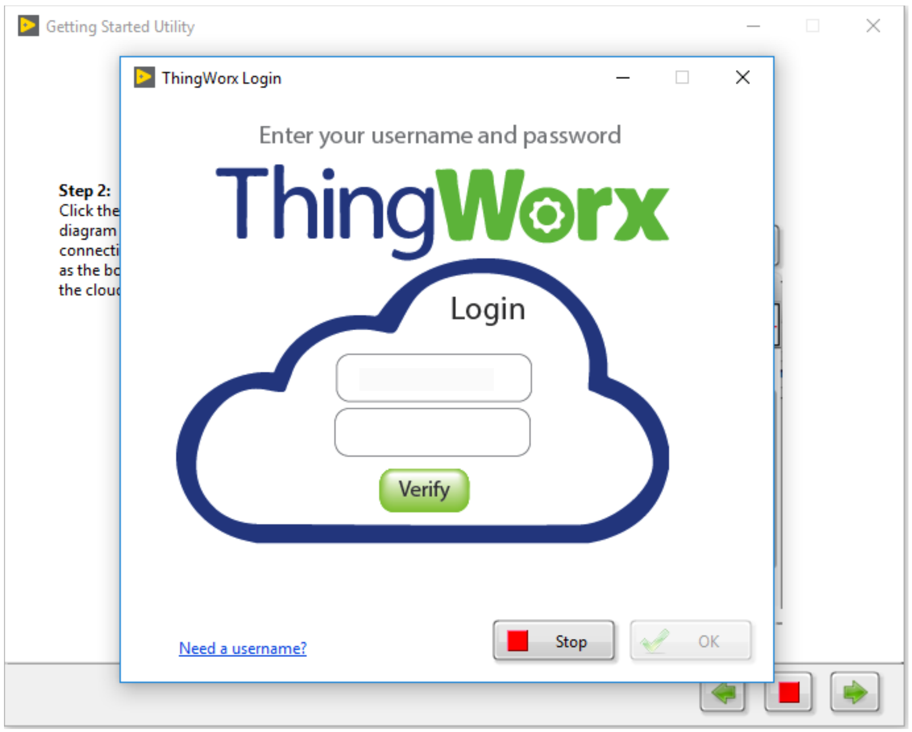
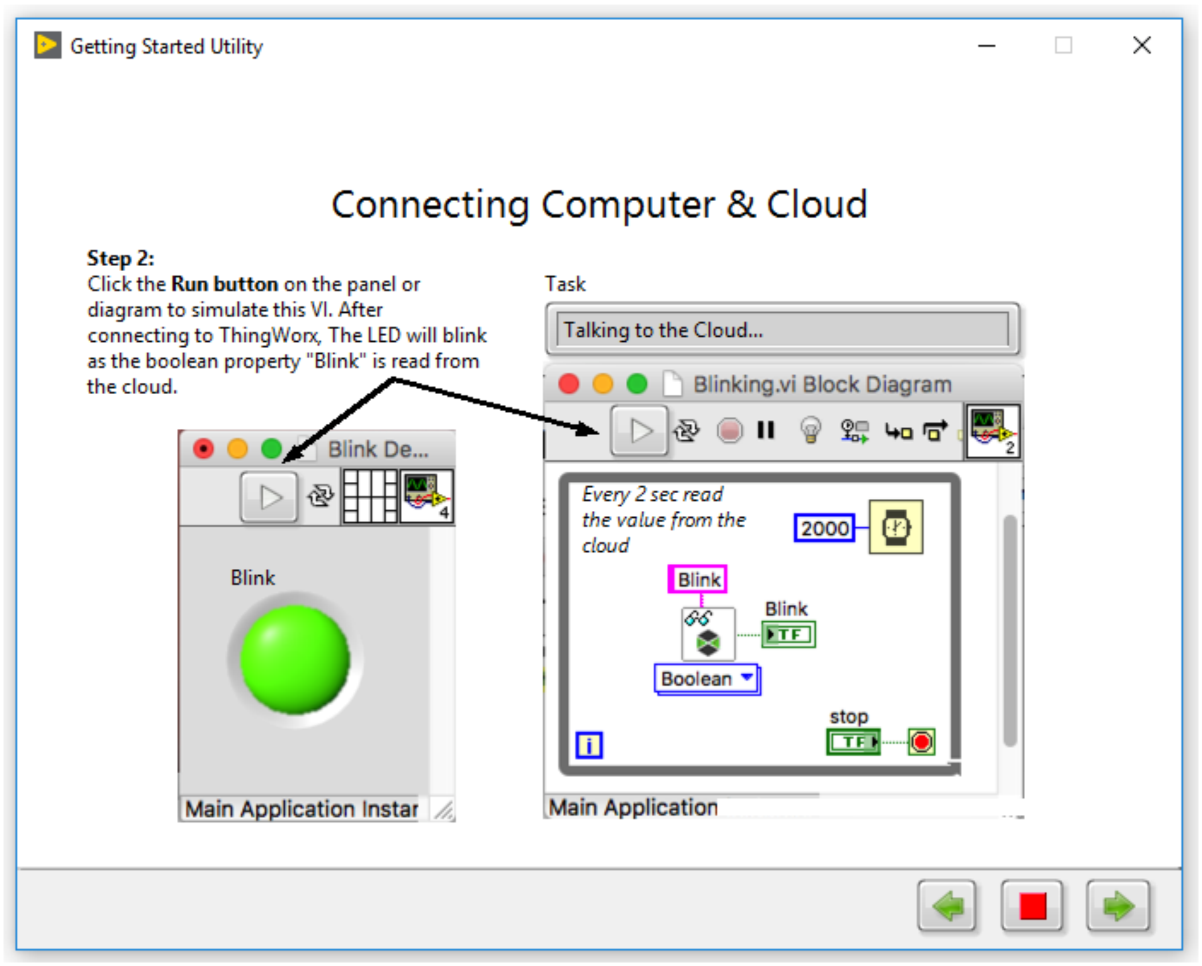


Figure 4.4: GSU: ThingWorx Login: Dialogue box for users to login to their ThingWorx accounts


Getting Started Utility

Connecting Computer & Cloud

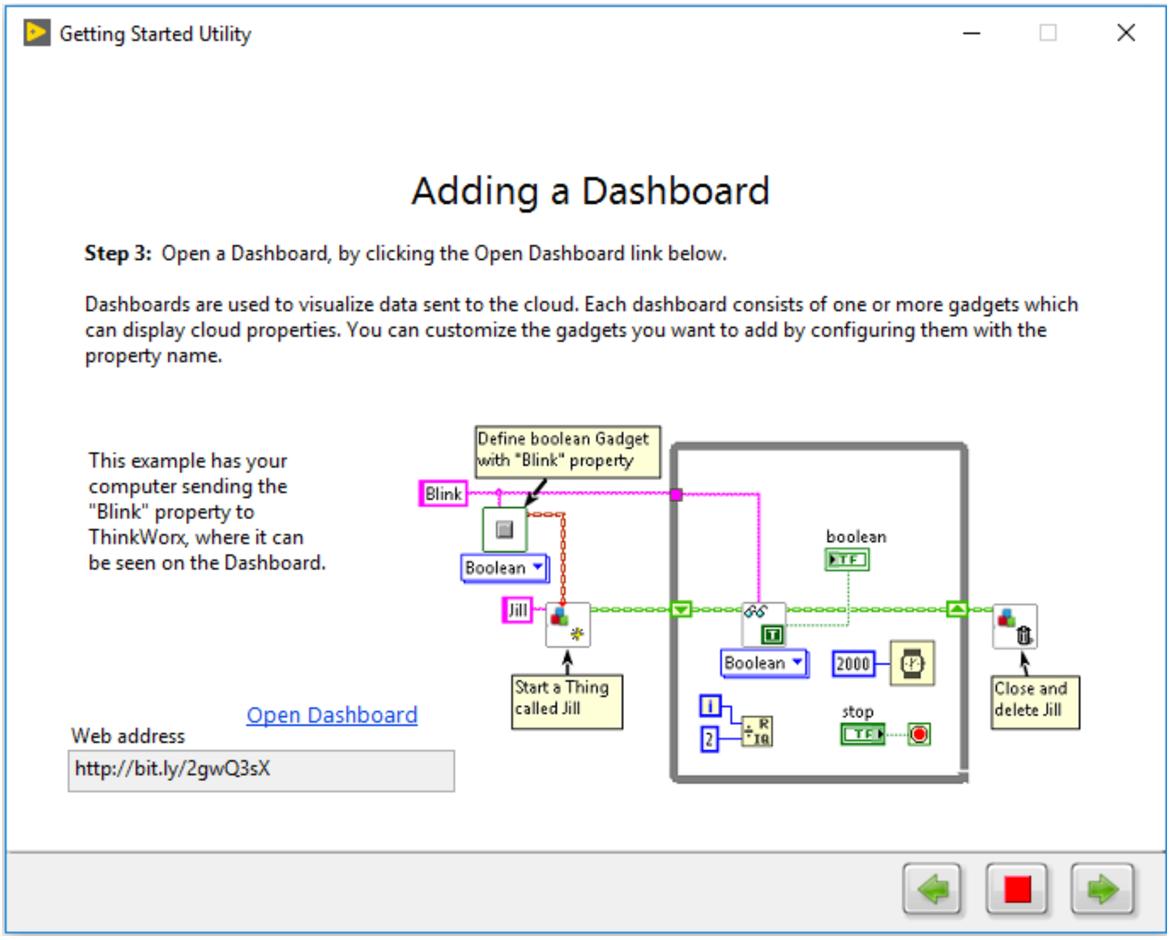
Step 2:
Click the **Run button** on the panel or diagram to simulate this VI. After connecting to ThingWorx, The LED will blink as the boolean property "Blink" is read from the cloud.

Task

Talking to the Cloud...

Figure 4.5: GSU: Connecting Computer & Cloud: A demonstration of the code running in the background of the VI that sends a boolean value from LabVIEW to ThingWorx, connecting computer and cloud.

The next step shown above in Figure 4.5 has a user connects his/her computer data to the cloud by sending a boolean (on/off) to the online ThingWorx dashboard from LabVIEW. Users can access their dashboard from the next slide (Figure 4.6) and see their boolean switch turn on and off as the switch almost simultaneously blinks from LabVIEW in the green LED light labeled “Blink” in the diagram above. These LabVIEW slides show the code that is running in the background of this program.



Adding a Dashboard

Step 3: Open a Dashboard, by clicking the Open Dashboard link below.

Dashboards are used to visualize data sent to the cloud. Each dashboard consists of one or more gadgets which can display cloud properties. You can customize the gadgets you want to add by configuring them with the property name.

This example has your computer sending the "Blink" property to ThinkWorx, where it can be seen on the Dashboard.

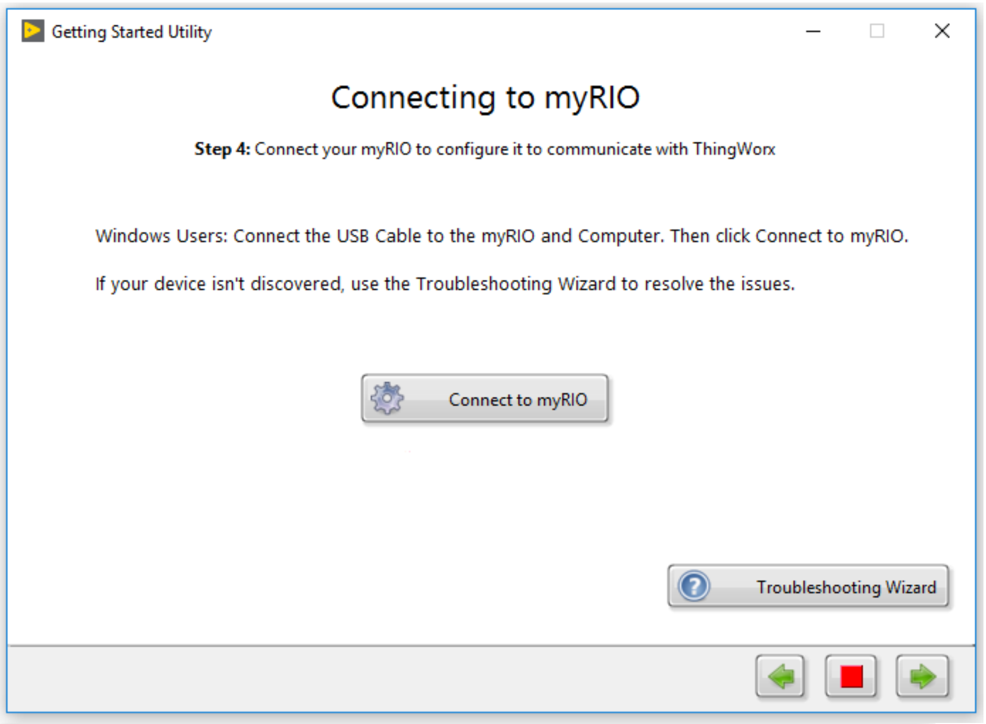
Open Dashboard

Web address: http://bit.ly/2gwQ3sX

Define boolean Gadget with "Blink" property
Blink
Boolean
Jill
Start a Thing called Jill
boolean
Boolean
2000
stop
Close and delete Jill

Figure 4.6: GSU, Adding a Dashboard: Shows users a version of code that creates a gadget on a dashboard and initializes a Thing with the associated property (a boolean in this case labeled “Blink”) under Thing “Jill.” Also gives the web address to user’s personal online dashboard to view this Blink property

The slide shown above gives the user a link to access his online dashboard. The code featured on this is page is from an earlier version of the Toolkit when it was still in early development stages.



Step 4: Connect your myRIO to configure it to communicate with ThingWorx

Windows Users: Connect the USB Cable to the myRIO and Computer. Then click Connect to myRIO.

If your device isn't discovered, use the Troubleshooting Wizard to resolve the issues.

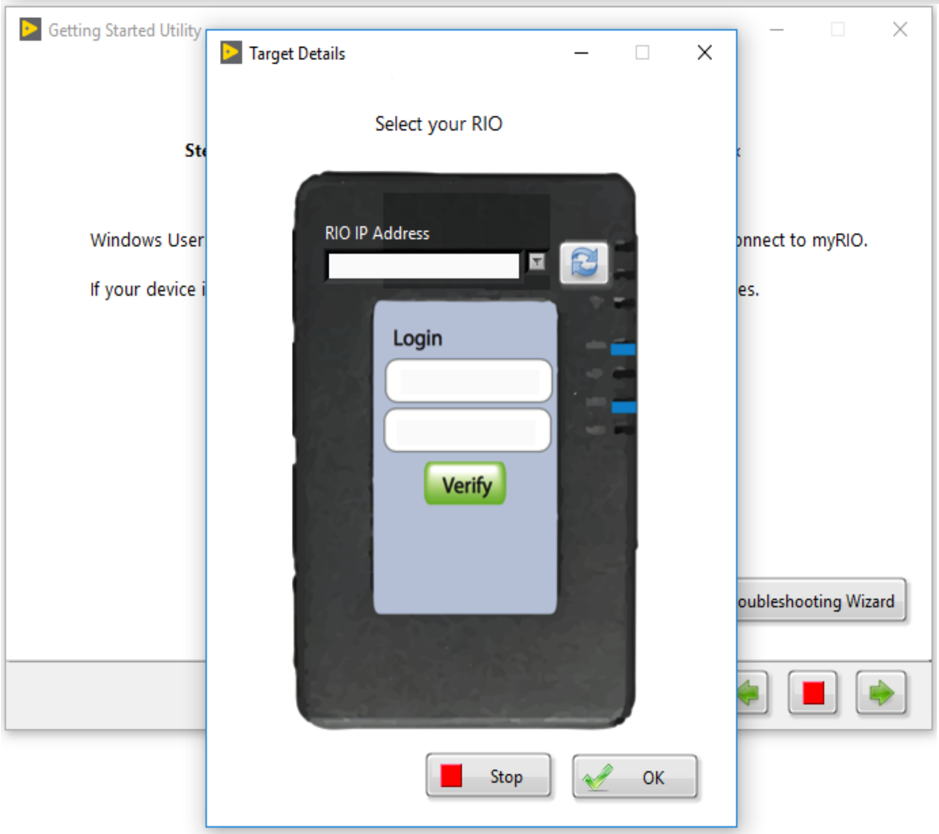
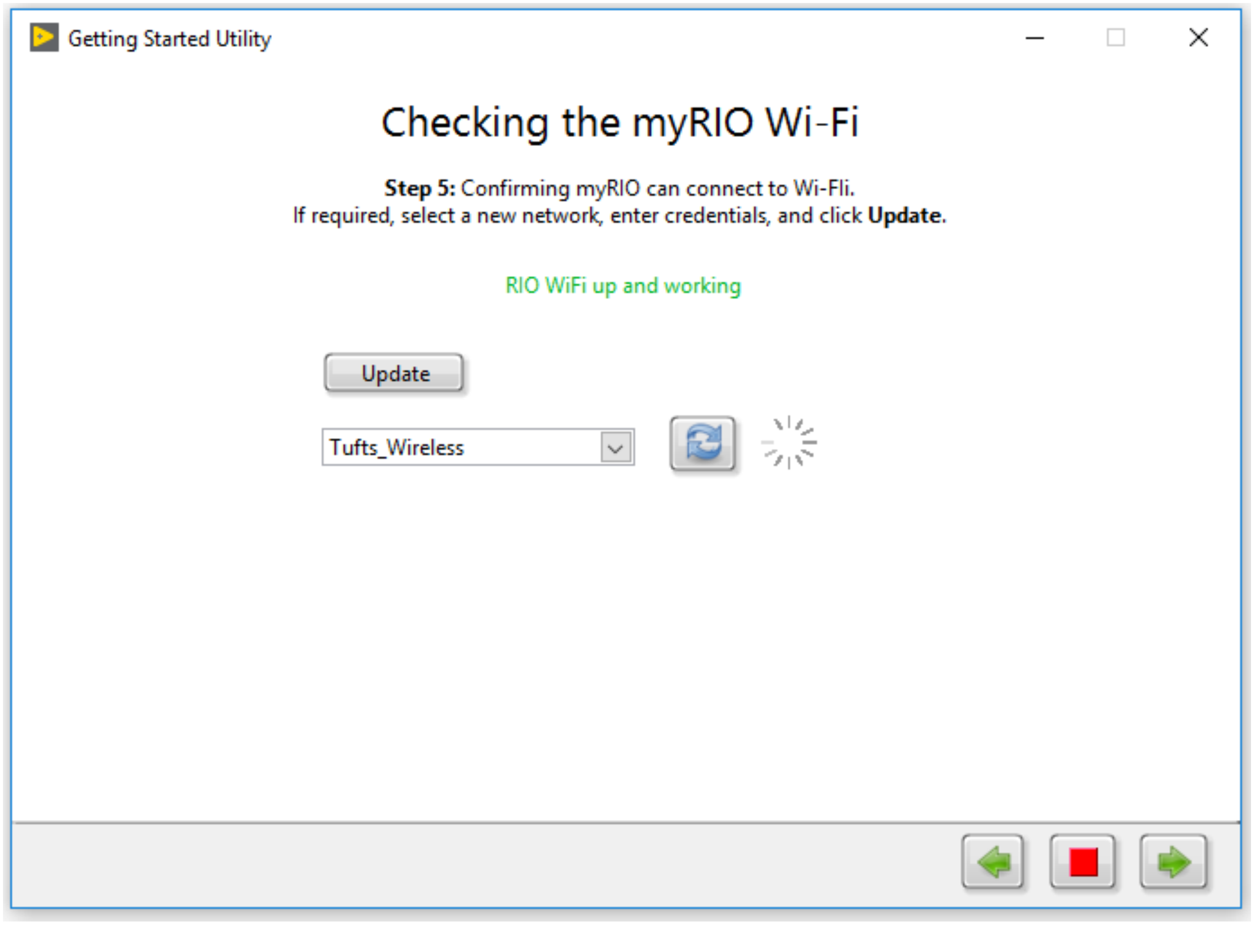


Figure 4.7: GSU, Connecting to myRIO: (Top) Explains how to connect to the user's myRIO. (Bottom) Gives dialogue box to enter myRIO credentials.


Getting Started Utility

Checking the myRIO Wi-Fi

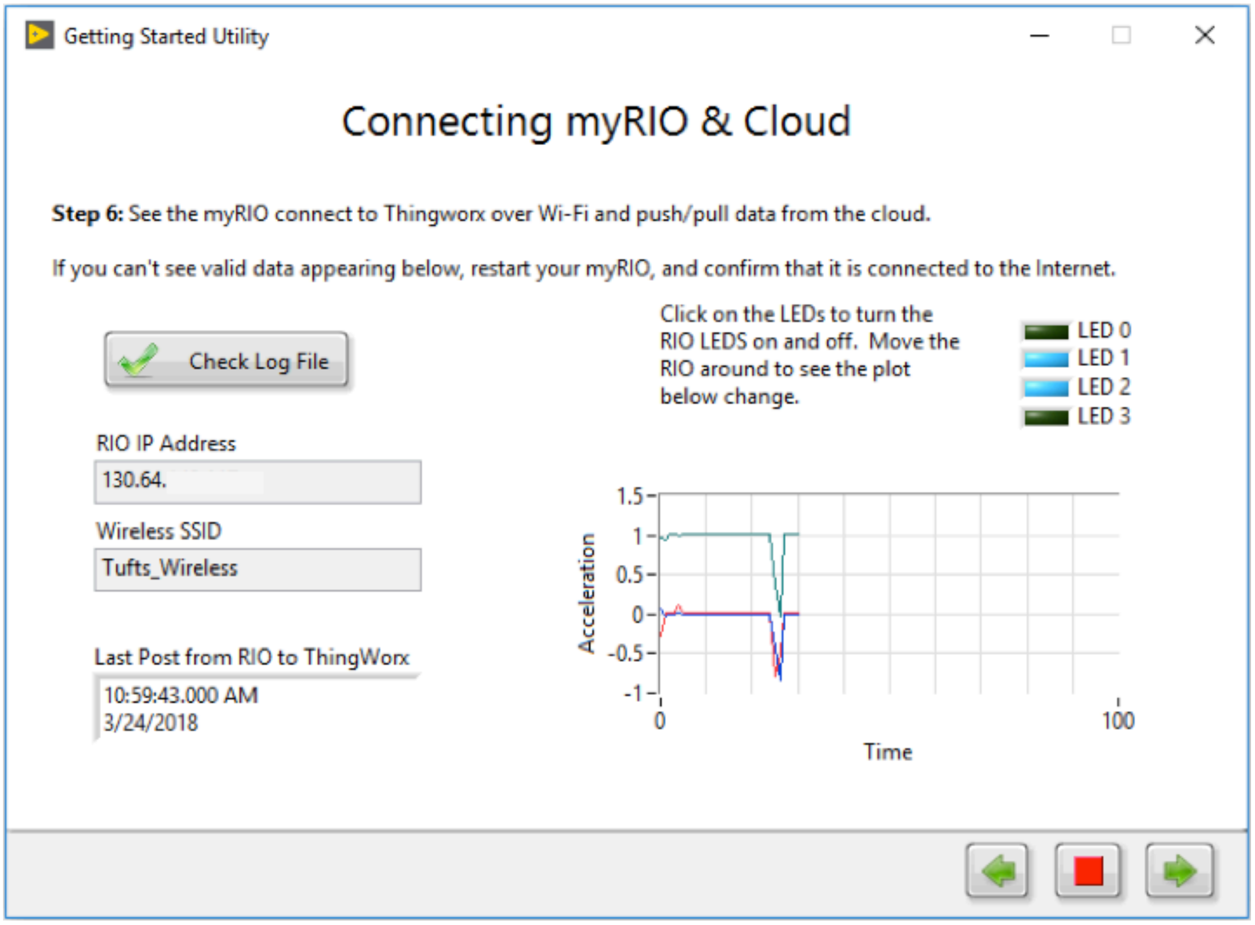
Step 5: Confirming myRIO can connect to Wi-FIi.
If required, select a new network, enter credentials, and click **Update**.

RIO WiFi up and working

Figure 4.8: GSU, Checking the myRIO Wi-Fi: Establishes a Wi-Fi network for the myRIO to run from

A user then adds the IP address of his or her myRIO. Users can either use the wireless IP addresses of their myRIOs, or they can use the 172.22.11.2 IP address to connect directly through a USB cord attached to the computer.

Then, users make sure their myRIO's are connected to the appropriate wireless network.


Getting Started Utility

Connecting myRIO & Cloud

Step 6: See the myRIO connect to Thingworx over Wi-Fi and push/pull data from the cloud.

If you can't see valid data appearing below, restart your myRIO, and confirm that it is connected to the Internet.

Click on the LEDs to turn the RIO LEDS on and off. Move the RIO around to see the plot below change.

RIO IP Address: 130.64.

Wireless SSID: Tufts_Wireless

Last Post from RIO to ThingWorx: 10:59:43.000 AM 3/24/2018

Figure 4.9: GSU, Connecting myRIO & Cloud: Shows a dynamic display of myRIO data that is traveling through the ThingWorx cloud to LabVIEW. A user can rotate a myRIO and see the Acceleration vs. Time graph change. A user can also select an LED button on this display to turn an LED on/off on the myRIO

The last step has users download the ThingWorx Toolkit software package to the myRIO to make sure the device can connect to the cloud properly and run the appropriate programs. To confirm the myRIO is connected, this page, shown in Figure 4.9, allows users to interact with the microcontroller to and from the ThingWorx cloud through this LabVIEW program by moving the myRIO and noting the change on the Acceleration vs. Time graph. A user can also turn the onboard LED's on the myRIO on and off through this program.

Once the user has connected his/her ThingWorx account and has initialized the myRIO, he/she can start writing code to send or receive to/from the cloud through the ThingWorx Express VI's I mentioned previously. The last page in the Getting Started Utility offers users resources to find examples.

Chapter 5 - Methodology and Procedures

I aimed to use a gap analysis approach to identify procedural and logistical issues in the current IoT Education Toolkit for future improvements of this system and future IoT education platforms designed for undergraduate engineering students. To detect these gaps I conducted interviews with two parties in comparison: National Instruments and PTC developers, and the target users: students studying within the engineering department who have recently used the Toolkit in an undergraduate classroom setting, or could conceivably do so in the future. I have added my own notes of the product's features, and perform usability testing with novel target users to further expose the product's strengths and shortcomings. I also conducted about a dozen unstructured usability tests with high schoolers, undergraduate interns at Tufts University, and students outside of the engineering department at Tufts. This informal feedback pointed to other flaws in the Toolkit, most of which also became apparent in the usability tests and interviews. Furthermore, I categorize these gaps as they fit into the five areas of focus, and suggest the next steps to converting the beta release into a more effective, efficient, satisfactory, and useful tool for the undergraduate engineering population.

5.1 Approach

As a method of determining the efficiency and effectiveness of the IoT education Toolkit, I have performed a gap analysis to measure the appropriate areas of improvement.

I have identified gaps along the five major categories of the Toolkit (getting started process, accessing and using the ThingWorx dashboard, debugging, adding complexity, and sharing data) through user interviews, usability tests, personal experiences, recommendations from other Toolkit users, and research of similar platforms that I have described earlier in this chapter. I have evaluated each of the categories through the following "gap" conditions:

1. Student expectations of the Toolkit
2. Student perceptions of the service
3. The actual service
4. Developers' (National Instruments and PTC) perceptions of students' expectations of IoT and the platform
5. Developers' translations of perceptions of user expectations into product specifications

I model these categories under Marketing Science Institute's "Gap Model of Service Quality" (Figure 5.1) (Carlson, 2010).

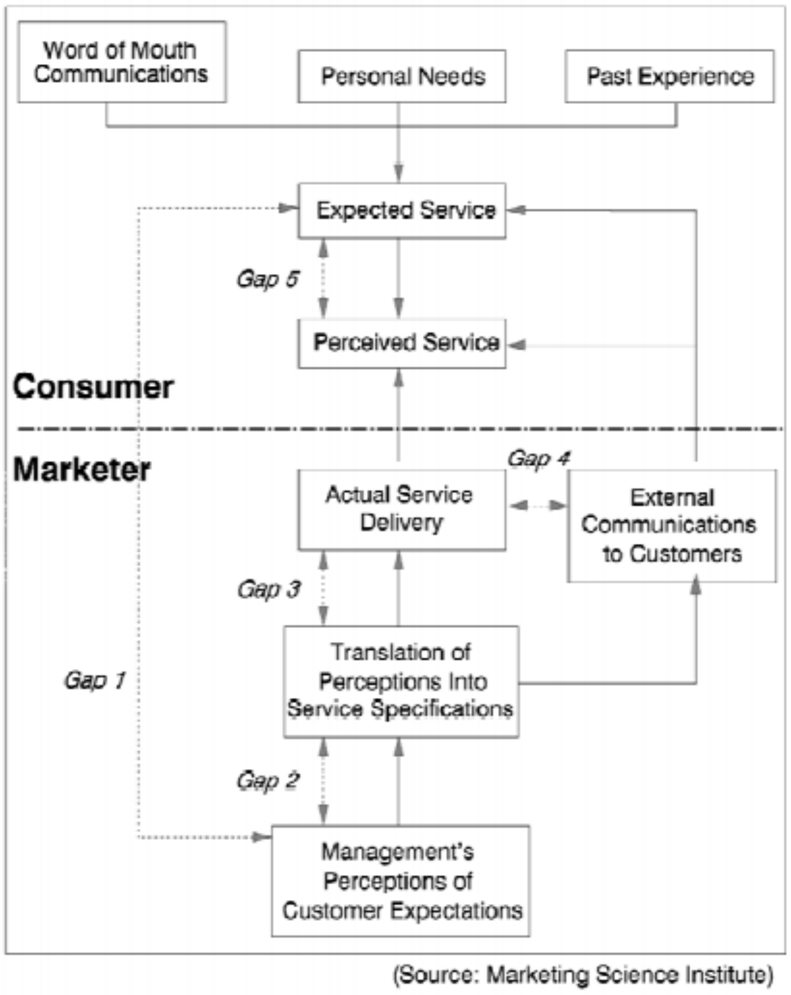

Word of Mouth Communications
Personal Needs
Past Experience
Expected Service
Gap 5
Perceived Service
Consumer
Marketer
Gap 4
Actual Service Delivery
External Communications to Customers
Gap 3
Translation of Perceptions Into Service Specifications
Gap 1
Gap 2
Management's Perceptions of Customer Expectations
(Source: Marketing Science Institute)

Figure 5.1: "Gap" Model of Service Quality: Compares products and services by aligning Marketer expectations and perceptions with those of the consumer. Can serve as a benchmarking tool to relate one company's product to another

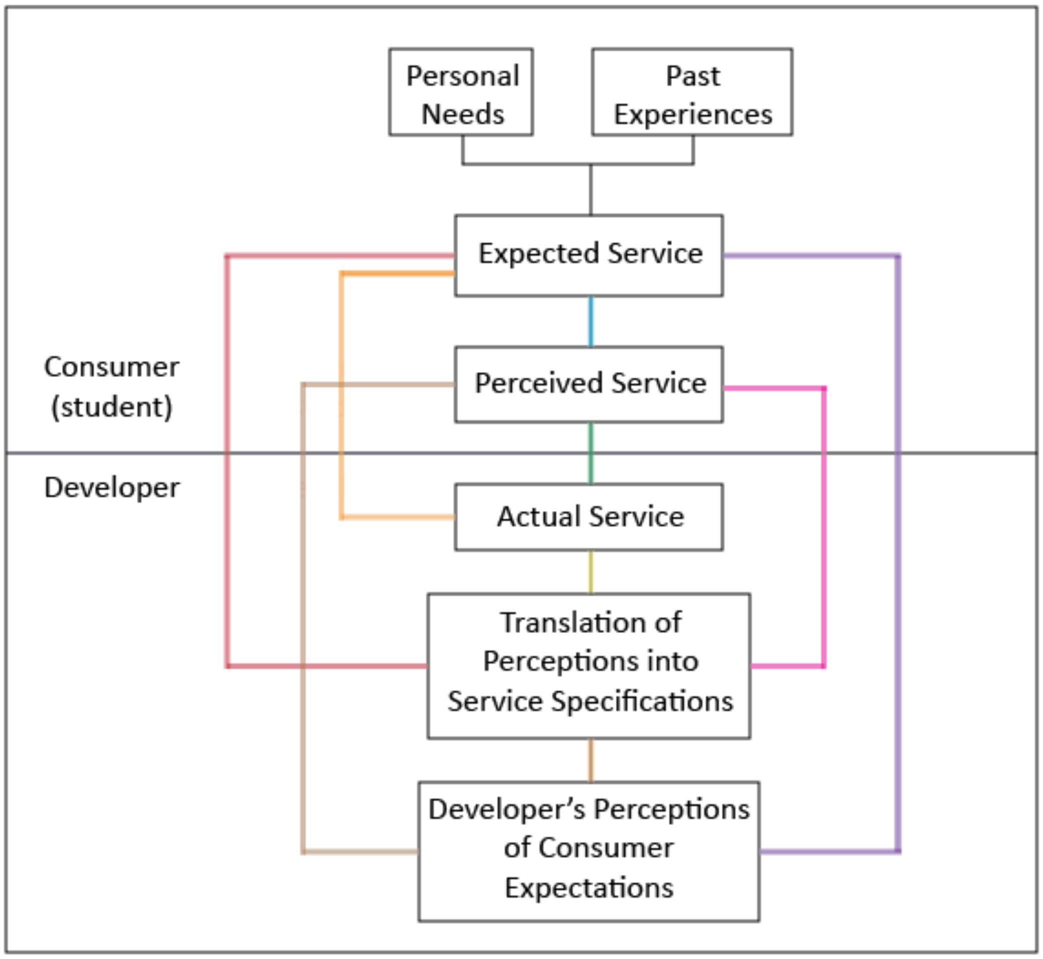


Figure 5.2: Marino "Gap" Model of Platform Quality: My modified version of Market Science Institute's "Gap Model of Service Quality" used to identify specific gaps of the IoT Education Toolkit Platform between developer expectations and perceptions as well as those of the consumer (student population). The colors distinguish different gaps between each region of perception or expectation

I exchange the term "Marketer" for "Developer," shown in Figure 5.2, to refer to the engineers and development team of NI and PTC developers since they were not necessarily trying to "sell" or "market" the product, but rather make it work well enough for the a beta version of the software for pilot users. The "consumers" in my model represent the mechanical or other engineering students at Tufts University.

According to my iteration of Marketing Science Institute's "Gap" Model of Service Quality (Figure 5.2), personal needs and past experiences all contribute to a student's expected IoT Toolkit /service. These inputs are acquired before the student begins working with the Toolkit. Personal needs includes the form of operating systems that students use (typically MAC, PC, or Linux compliant laptops). The IoT Toolkit software will differ slightly based on which computer type the user has. In order for the platform to be most convenient for the student, it must be adaptable for each computer type. Personal needs can also include individual academic interests of the student, like learning about robotics or IoT subjects. Past experiences can include but are not limited to computer programming, robotics, and other IoT platform experience. All of these details contribute to the user's expectations of the platform before or as they begin trying to send data to the cloud.

The expected service refers to how students anticipate the features of the Toolkit to work based on their personal needs and past experiences, and the perceived service is that in which they actually interact with. This version of the service and its features can differ however from the actual service, or the Toolkit in its entirety. The actual service may include features that users did not know were available, for example.

When beginning to create the Toolkit, the developers formed perceptions of how students anticipated an IoT Education Toolkit would work and what features it would include. They then translated these perceptions and created service specifications for the Toolkit. These specifications would form the building blocks of the actual service.

5.2 Interviews

Interviews provide flexible means of gathering specific information regarding a particular subject (Stanton, 2017). I performed semi-structured interviews with the NI and PTC developers to help reveal the thought processes and motives that drove the creators of the platform into organizing the Toolkit as such. I also ran user interviews with Tufts University students who used the Toolkit in a robotics course from the prior semester. By semi-structured, I refer to the interview method of asking the participant pre-determined questions as well as unplanned questions that arise within the session.

5.2.a. NI &PTC

I facilitated interviews with the two developers over Cisco's WebEx, an online meeting space that allows meeting recordings, which I used during both sessions. I also typed notes during the sessions on my laptop. Both of these interviews lasted no more than one hour. I asked the participants specific and general questions about the Toolkit regarding the getting started process, the online dashboard, debugging, adding complexity, and sharing data. Through this approach, I helped identify the developer's perceptions of user expectations, the translation of these perceptions into service specifications, and the actual service delivery (Carlson, 2010). These classifications help to identify the gaps that emerge within the product development team and between the product and the consumer, who are, in this case, the students using the platform.

Although I only received feedback from two developers, one from each of the different contributing parties from industry (NI and PTC), I considered their responses representative of their associated departments' opinions and understandings. The developer from National Instruments is a software group manager who helped to mold the platform into its "shippable" form. He would make architectural changes and fix bugs to help make the Toolkit run smoothly for users. The other member of the developing team I interviewed is a PTC employee who is the group lead of the academic technology group. He served as the lead engineer of the ThingWorx components of the Toolkit, working on the cloud aspect of the IoT solution. He and his team aimed to take the IoT ThingWorx platform, meant for many industrial IoT solutions, and add additional scaffoldings to allow novice users an easier approach to create their own IoT projects in a the classroom. Both of these interview participants stated that the target users were students at the university level who may or may not have some technical knowledge (in coding and/or IoT), or are studying a course that involves electronics. Both of the interview participants were involved in the early stages of development, giving constructive contributions and feedback based on their experiences.

5.2.b. Undergraduate Robotics Students

User interviews consisted of no longer than one hour sessions with five students from an undergraduate senior robotics class from the Fall semester of 2017 at Tufts University. This class contained sixteen students from the Mechanical Engineering department. I held the sessions at on-campus locations at the Tufts University Medford Campus and audio recorded the responses as well as took real-time notes on my laptop. I asked questions on their opinions and usage of the Toolkit/IoT and specific experiences in the five aspects under discovery. These questions helped identify the consumer's expected service and perceived service. The interviews also defined the students' background in robotics and IoT as well as research interests as engineering students.

User profile

The five students I interviewed were all pursuing degrees in the Mechanical Engineering Department at Tufts University. Four of them were students in their fourth year of study and one was a second year Master of Science student/Ph.D. candidate. Three of the five participants had at least two semesters worth of LabVIEW experience, and the other two had no or "minimal" experience.

5.3 Usability Tests

After the user interviews, I performed the usability tests with users in the appropriate population according to National Instruments' and PTC's assumed user population, namely, undergraduate and graduate students studying engineering/robotics. I asked students from Tufts' Mechanical Engineering Department to participate and give feedback. The tests were about one hour long, and I designed the sessions to test basic Toolkit functionalities with novice users. These users do not include students who participated in the Fall 2017 robotics course.

User profile

The eight participants, three male and five female, were students at Tufts University in the Mechanical Engineering Department, either studying mechanical engineering, human factors, and/or STEM (science, technology, engineering, mathematics) education. The ages of the participants range from 21-32. Two students were undergraduate students, and 6 were graduate students either pursuing a master's of science or Ph.D.

These participants all study within an engineering field at Tufts University. The participants studying within the Human Factors department are part of the School of Engineering. They all have never used the IoT Education Toolkit, and have varying coding and LabVIEW experience. These novice users are therefore ideal usability participants for this test, since they could conceivably encounter an IoT development application in their academic careers while developing themselves as engineers.

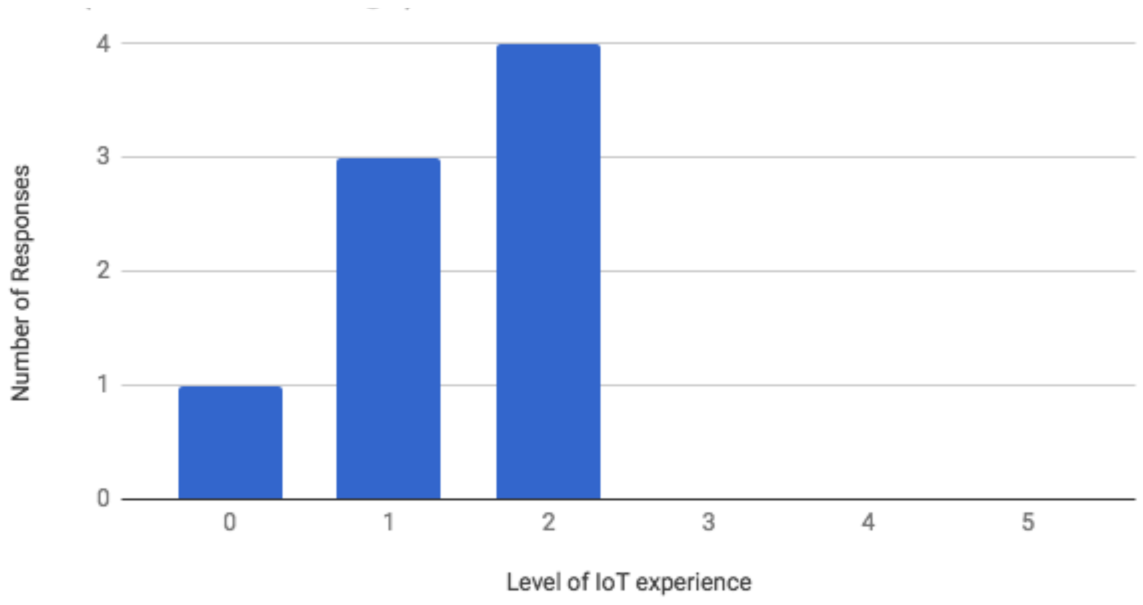


Figure 5.3: Usability Test Participant Response to Question: *How much experience do you have with IoT (Internet of Things)* 0 = I've never heard of IoT, 5 = I've comfortably developed several or many IoT projects

The above chart (Figure 5.4) shows that although most users had at least heard of IoT and had some experience, none of them had comfortably created more than a few IoT projects, if any.

5.3.a. Pre-Test

Before the usability test, I asked each participant to sign a consent form and complete a pre-test survey to determine basic academic information, LabVIEW/coding skills, and IoT technology backgrounds.

5.3.b. Test

To capture the users' work, I recorded the screens of all of the participants in the study as they were working. The screen recordings featured audio recording as well so I could hear the users as they thought aloud. Most of the participants did not think aloud however. I lost two of the user recordings due to technical issues, but later had the respective participants recount their processes, code, successes, and mistakes, in retrospective interviews. I conducted one of these interviews immediately after a participant needed to end his usability test session. I was able to audio record this interview. The other retrospective interview occurred within six days of the usability test. I was able to show this participant her code, and ask her similar questions about her thought processes, logic, and reasoning behind her code. I audio recorded this interview as well.

The goal of the usability tests was to measure how well novice users could navigate through the system to create basic IoT solutions, and how well they understood how data was flowing through the systems they were creating. I gave each participant a laptop, myRIO, and Hue Lights Strip, each labeled with their assigned “shape” as a name (Triangle, Rectangle, Circle, or Square) so users could easily identify which tool was theirs. Hue Lights are Internet enabled IoT lights that owners can adjust using a mobile application (Philips Lighting Holding B.V, 2018). I created a VI in LabVIEW to manipulate Hue Light Strips’ color, state, and brightness so users could alter these values through manipulating property values through the Toolkit Express VI’s.

Participants’ Hue Lights Strips were all turned on before the test. After each participant signed a consent form and completed the pre-test survey, I had them watch a five-minute video that quickly explained IoT, LabVIEW, ThingWorx, and the myRIO. (Appendix A)

I then handed out directions (Appendix B) for four tasks. I told the participants that throughout the usability test, they could refer back to the video they had just watched, or any Internet source for additional help. In an Internet browser on their assigned laptops, I had open the online ThingWorx dashboard, the instructional video I had shown them, and the post-test assignment. I also provided a sheet that offered additional LabVIEW tips (Appendix C) for notice users to the coding platform. This handout, labeled “Helpful Tips,” outlined necessary features to complete the tasks, including how to reach the Toolkit’s Express VI’s, how to create constants, controls, and indicators, how to create a while loop, and how to find the myRIO button block. I included these tips to save time by eliminating the need to search for LabVIEW help instead of focusing on completing the tasks. I told participants that they could ask me questions, especially about LabVIEW coding, but that I might not answer all questions so I could see how they worked through them.

I also created two LabVIEW projects for the participants before they began the tasks. One project would run the code from the laptop, and the other from the myRIO. In LabVIEW, users can run code from two different machines on the same project, but I decided to organize the two computers (laptop and myRIO) into two separate projects so novice users could easily distinguish which machine they were running code on (see Appendix B for visual).

Measures of Success

The four tasks in the usability test aimed to gauge how efficiently and successfully users can operate the system to develop simple IoT solutions, and then measure how well they understood their system's data structure in a post-test assignment. I measured their success of each task based on if their code ran properly, if it was organized similar to how I instructed in the Task Handout (Appendix B), and if they completed acceptable post-task assignment diagrams similar to how I outline them (see post-task assignment descriptions below). I illustrate further acceptable measures for each task below.

Task 1: *Write to the Thing called "Hue lights_yourassignedname" from your computer. Change the property color from the front panel in LabVIEW. Make sure you're running from your laptop.*

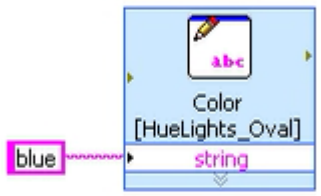


Figure 5.4: Usability Test, Task 1 Code: A “write” Express VI under Thing “HueLights_Oval.” Changes the string property value to “blue,” which changes the Hue Lights Strip lights to this color

I considered the participant to successfully complete Task 1 when he created a coding diagram similar or identical to the one shown above (Figure 5.5), and was able to turn his Hue Lights Strip a different color. The user would also have to be running his code from his laptop project in order for it to count as a success. He must have a “write” Express VI that controls the color of his assigned Hue Lights Strip.

Task 2: *Make the button on the myRIO change the state (on/off) of your Hue Lights Strip. Make sure you create a while loop around your code. Make sure you’re running from your myRIO.*

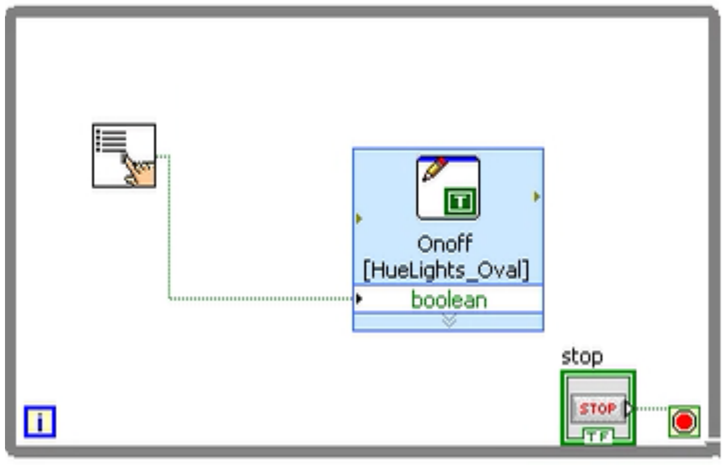


Figure 5.5: Usability Test, Task 2 Code: A “write” Express VI that denoting Thing “HueLights_Oval” and property “OnOff.” The myRIO button value controls the boolean value of the Hue Lights Strip state.

A user would run Task 2 successfully if she switched from the Laptop project to the myRIO project and created a code similar to or identical to the one shown above (Figure 5.6) (I showed participants how to find the myRIO button indicator within the Toolkit in the “Helpful Tips” handout, but they can also find a similar VI under the myRIO library in LabVIEW, which would be acceptable to use as well. The user would also have to successfully press the button on the myRIO and turn her Hue Lights Strip on or off. She must have a “write” Express VI that controls their assigned Hue Lights Strip.

Task 3: *Pull the property “Next Bus” and control the color, brightness, or state (on/off) from the myRIO based on how close/far away the next bus is.*

This property gives you the time of the next MBTA 94 bus leaving from Boston/College Ave going toward the CEEO (a laboratory on Tufts University Campus)*, or vise versa.*

Brightness range is from 0-254. Turning the brightness to 0 will turn the Hue Lights off.

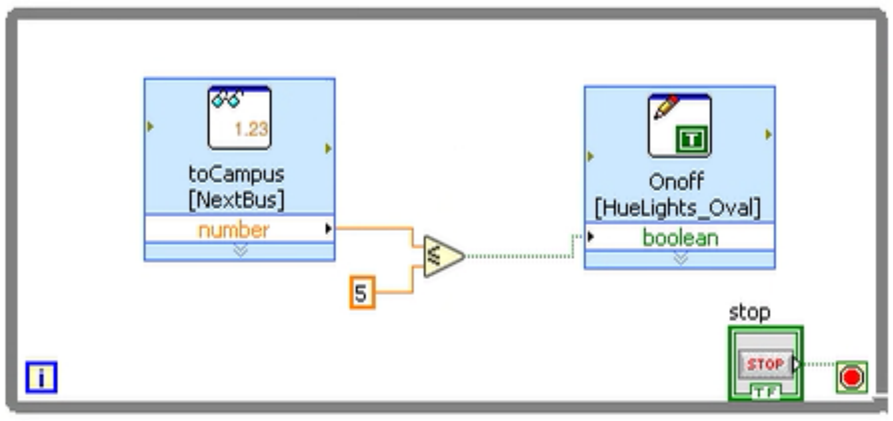


Figure 5.6: Usability Test, Task 3 Code: A “read” Express VI that receives a number from the cloud that gives the minutes until the next bus arrives. Logic is added so when the bus is five or fewer minutes away, the Hue Lights Strip turns on.

Users must be running their code from the myRIO project, but this task has some more flexibility for success compared to the first two. Users are given more freedom to use coding logic within LabVIEW to have the Hue Lights change based on a changing variable. An example of a successful code is shown above (Figure 5.7), but there are many options for this task to constitute as a success. They must have included a “read” Express VI that reads one of the Next Bus properties, as well as a “write” Express VI that controls a property in their assigned Hue Lights Strip. These VI’s must be connected somehow in LabVIEW. If the light strip changes brightness, state, or color based on a Next Bus property, then the user has successfully completed this task.

Task 4: *Simultaneously, have the computer control the color of your light strip and the myRIO control the brightness or state (on/off)*

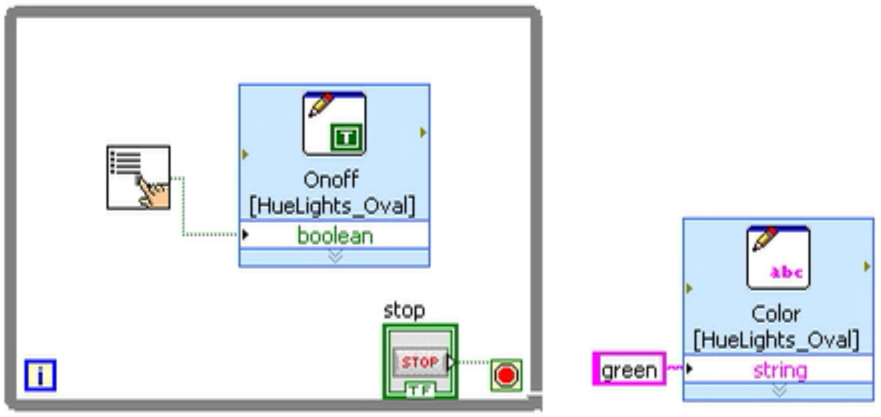


Figure 5.7: Usability Test, Task 4 Code: (Left) A “write” Express VI that reads the boolean value initiated by the press button on the myRIO. (Right) A “write” Express VI that changes the color of the Hue Lights Strip. The figure on the left should be run on a myRIO project VI. The Express VI on the right should be on laptop/computer, according to the task’s directions.

Here, the user must be running code from both the laptop and the myRIO projects. The code on the laptop must have a “write” Express VI that controls the property “color” of the assigned Hue Lights Strip. The myRIO project should have code that controls the brightness or state property of the light strip. If the code signifies these relationships and the color and brightness/state of the Hue Lights Strip change respectively, then the user completed the task successfully. I can check these values in ThingWorx Composer, and the user can also check his or her code via LabVIEW indicators to make sure the data is running properly.

5.3.c. Post-Test

I told the participants that the usability test would be one hour in length, so if they were not finished completing the tasks within about 45 minutes past the start time, I asked them to stop where they were and begin completing the post-task assignment and post-test survey if they needed to leave the study after one hour.

Post-task assignment

Rebecca Lawson, a Psychology Scientist at the University of Liverpool, used an objective method to determine how people understood how bicycles work (Lawson, 2006). She presented several graphics to the participants (a bicycle separated from its petals, frames, and chain), and asked them to organize given symbols to fill in the main bits of the frame of the bicycle that the participants thought were missing. I used a similar approach in the post-task assignment to determine how well the participants understood how data was flowing through the IoT system based on the information I gave them.

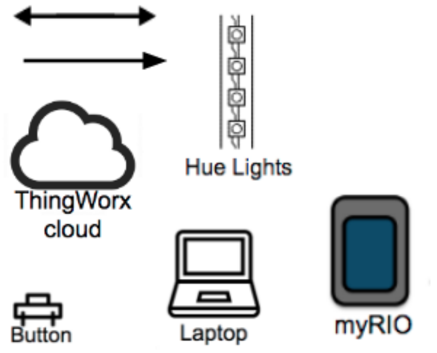


Figure 5.8: Post-Task Assignment Graphics: Users were asked to demonstrate how data was flowing through their IoT systems using these graphics

I asked the participants to illustrate how data was traveling through the system using the graphics provided in Figure 5.9, or by creating their own using objects with labels and/or descriptions. I reminded them of the task assignment and provided them with the icons and graphics above (Figure 5.9)

I told them they could move the graphics as they see fit, and they would add graphics, delete them, copy and paste, etc. I told participants to complete at least the diagrams for which they completed the tasks for, but if they felt comfortable, they could fill in the diagrams for the other assigned tasks.

The following diagrams are examples of “successful” post-task assignment diagrams.

Task 1

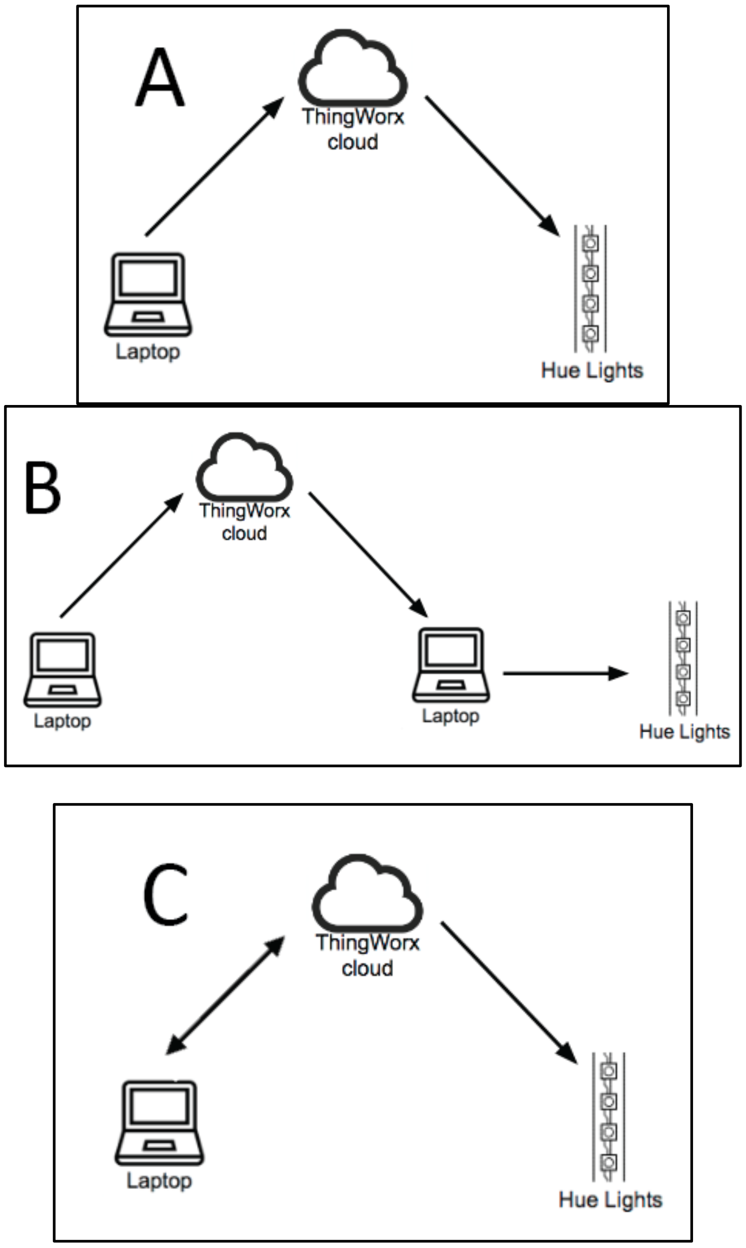


Figure 5.9: Post-Task Assignment #1: Examples of correct post-task diagrams for Task #1, where users were asked to change the color of the Hue Lights

Figure 5.10 (B) shows a laptop controlling the lights instead of directly from the ThingWorx cloud. Any of these versions I considered correct. Before testing, I mentioned briefly to each participant how I was running code from my laptop that was reading all of the Hue Lights properties and writing values to control the Hue Lights. However, I accepted both diagrams above, since I did not present this information clearly. Throughout the rest of the following “successful diagrams,” I did not include the laptop diagram controlling the Hue Lights Strip, but if a participant included the second laptop in his or her system controlling the lights, I measured it as a success.

Figure 5.10 (C) above shows a double-sided arrow connecting the laptop computer with the ThingWorx cloud. Although the computer may not be writing any data, the computer still receives feedback from ThingWorx. Without written or verbal descriptions or labels it is impossible to determine if the participant intended for the arrows to represent read/write command paths, as they were designed for. Therefore, I considered the diagram to be representative of the data flow if the participant decided to include a double-sided arrow from laptops to the ThingWorx cloud, since they can access the cloud data through the ThingWorx dashboard from their laptops, and from the Express VI windows, presented on the laptops.

Task 2

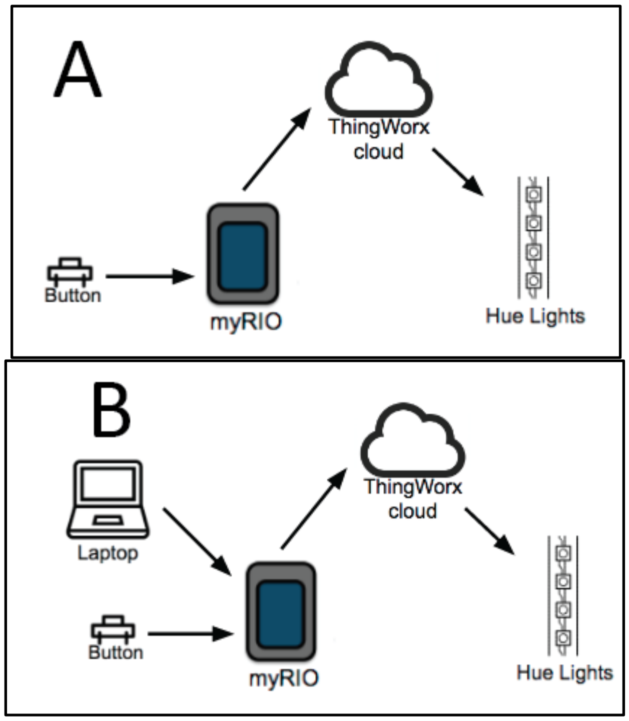


Figure 5.10: Post-Task Assignment #2: Successful illustrations representing data flow diagrams for Task #2, where participants were asked to run a program from the myRIO to change a Hue Lights property value when the myRIO button was pressed

In this task, the user should have been pressing the button on the myRIO and having the button boolean value (on/off or true/false) control the state of the Hue Lights Strip via the ThingWorx cloud. Technically, the laptop controlled the code that is running on the myRIO, so if a usability participant included an addition of this laptop to the myRIO, as shown in Figure 5.11 (B), I assumed that they are signifying this process and measured the system as a correct diagram. This is true as well for task 3 and 4, which use the myRIO.

Task 3

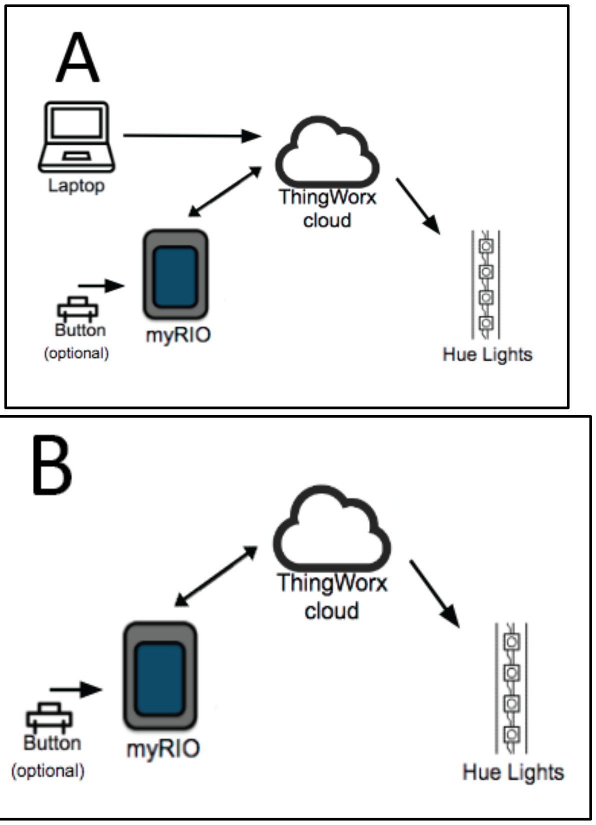


Figure 5.11: Post-Task Assignment #3: Two versions of successful user diagrams showing the data flow for Task #3, which asked participants to read a NextBus value and control the Hue Lights Strip from the myRIO

Here, the double arrows represent the myRIO reading and writing properties to the cloud. In Figure 5.12 (A), the laptop is writing values from the Next Bus to the cloud, and the myRIO is reading these values, and writing properties to the cloud to control the Hue Lights. Users could use the button to control these output values, but they did not need to. I mentioned before the usability test that the laptop was pulling the Next Bus values from a 3rd party source, and writing them to the cloud, but participants did not need to include this in their diagrams since I was not clear about where these values were coming from.

Task 4

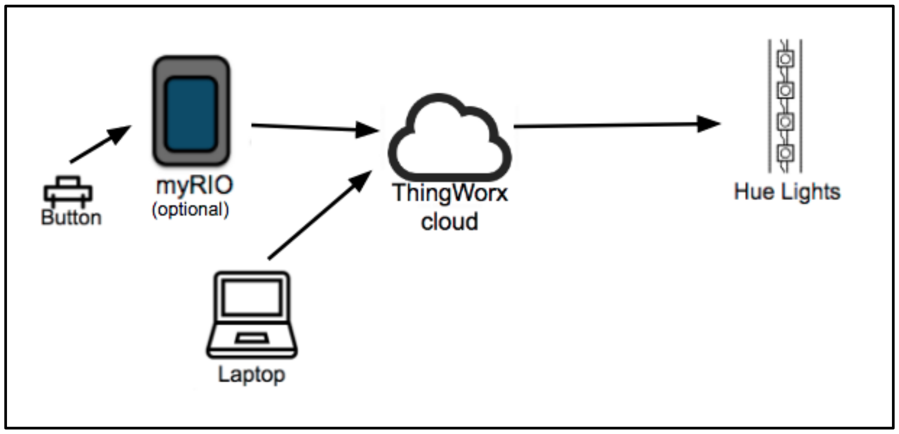


Figure 5.12: Post-Task Assignment #4: Data flow diagram representing Task #4's data system, controlling different Hue Lights property values from a laptop and myRIO

The final task asked users to write to the cloud to control the Hue Lights from both the myRIO and the Laptop.

Post-test survey

I also asked participants to complete a post-test survey once they were finished with the post-task assignment. This survey helped to further determine user's opinions and understandings of the Toolkit.

Chapter 6 - Results

In this section I present the results from the interviews with students and developers as well as the student usability tests. The Appendix D gives a detailed summary of the results of the usability tests. I organize the results into the five discussed areas of the Toolkit (getting started, ThingWorx dashboard, debugging, adding complexity, and sharing data).

6.1 Getting Started

According to an interview with a National Instruments developer, not all students are expected to have experience with IoT or LabVIEW in order to begin using the IoT Education Toolkit. They will need some preparation from classroom professors or teaching assistants, but the platform should be accessible to novice users.

The "getting started process" I will refer to as the steps users take after they have correctly downloaded the appropriate software on their laptop computers or desktops, and when they begin to run the Getting Started Utility to initialize their ThingWorx accounts and myRIO's. I only include feedback from the Getting Started Utility from the student interviews from the Fall 2017 robotics course who had experience with this feature during class. I also include in this section the first instance a user sends and receives data from the cloud, which is completed in usability testing.

6.1.a NI/PTC Interviews

This Getting Started Utility has many obvious areas of improvement. Every user needs to go through this setup program in order to initialize ThingWorx accounts with the computer and the myRIO. Both the NI and PTC developers agreed that this step should be as simple as possible, with minimal user involvement. Some screens in the program however show some LabVIEW code, since originally, developers were hoping to convey some helpful LabVIEW information for users to get exposed to some of the code that is running in the background of the program. Figure 4.6: GSU, Adding a Dashboard, for example, shows code that could create dashboards from the LabVIEW block diagram. Developers figured that this might have been useful for users as they began writing code to send to ThingWorx. The NI developer stated that there were hopes that students could begin using this platform with no LabVIEW experience. However, this developer also expressed in the interview that after some informal user testing before the system's beta release, that users did not take the time to comprehend the LabVIEW code diagrams, or read the descriptions or captions that the screens offered. Furthermore, the displayed code is outdated from the beta release version, as developers decided to use the Express VI's (shown in Figure 2.2: LabVIEW IoT Education Toolkit Express VI) instead of the more elaborate LabVIEW code featured in this Utility.

In summary, the developers' goal was to build a system that would allow users to create their own IoT solutions for the first time using this platform, even if they had no previous exposure to LabVIEW or IoT.

6.1.b Student interviews

Moreover, user interviews from students who completed the robotics course in the Fall 2017 semester further revealed that the getting started process was a “high barrier” to entry for novice users, even though the process was fairly linear. One user stated, “The getting started barrier is so high… To understand what those (Express VI’s, 3rd party VI’s, example VI’s, etc.) are actually doing takes so much time, and I think that’s the hardest part about the Toolkit.”

Table 6.1: Student Interview: Getting Started

Question/Topic	Responses
Expectations with the getting started process (Including the Getting Started Utility and sending/receiving data to/from ThingWorx for the first time)	“The two platforms should be able to talk, so I assumed that [the GSU] would download some sort of drivers”
	“I assumed that [the GSU] would show me an example that I could use to then build whatever I need to build.”
Problems or confusions with the getting started process	“There were a lot of issues with having the latest image (software) and those images maybe weren’t compatible with the LabVIEW version that you had, and even if your LabVIEW version and the ThingWorx image were right on your machine, then they might not be right on the myRIO, and especially since we were switching partners every week, it meant that we were switching myRIO’s, and so you have it right one week and have to uninstall and reinstall for next week, and that process was frustrating. You want to be able to plug it in and know that everything that’s installed is correct.”
	“Honestly, at first it was a little confusing on what this cloud feature does... I like to see Things visually, so for me I would’ve liked to see a demonstration.”

	"People don't always read, and so if you're new to this, you're opening everything up, and you're just clicking through, then you might not get that understanding that Things are being sent and received via the Internet... I'm not sure everybody really conceptualized that from the Getting Started [Utility]."
	"It wasn't as clear as possible on what you should be doing [in the GSU]."
	"I'll admit, I'm not too good at reading directions… I'm just more of a 'on the go,' so for the myRIO, it was more like me plugging in and hopefully that it'll prompt me to do the next step."
	"If something goes wrong, it's hard to figure out what went wrong."
	"We had a lot of times where the Internet wasn't set up correctly on the myRIO, so you thought you were pushing to the cloud, but you weren't."

The robotics students understood the "getting started process" as navigating through the Getting Started Utility. These responses reveal some issues regarding the myRIO as it related to the GSU. All of these users expressed the lack of their understanding, or that of other students in the class involving what the Getting Started Utility was actually doing in relation to data flow and installations. They do not refer to specific user interface pain points, perhaps because they had not used the Utility in several months, but reflect on specific areas of the Toolkit as they remember them in terms of installations and explanations of data flow.

6.1.c. User Testing

Users did not go through the Getting Started Utility during testing, because of a time constraint with user testing, and since the student and developer interviews revealed many if not all of the apparent gaps, but I include in this “getting started process” the procedures of connecting the myRIO to the computer, and sending data to the cloud for the first time using the Express VI’s.

The pre-test survey measured that half of all the usability participants had taken a robotics course during their university careers, but all participants reported that they have had little to no experience with IoT (rating themselves 2 or below on a 0-5 scale, 0 = “I have never heard of IoT, 5 = “I have comfortably developed several or many IoT projects). A summary of other key findings is shown below in Figure 6.1.

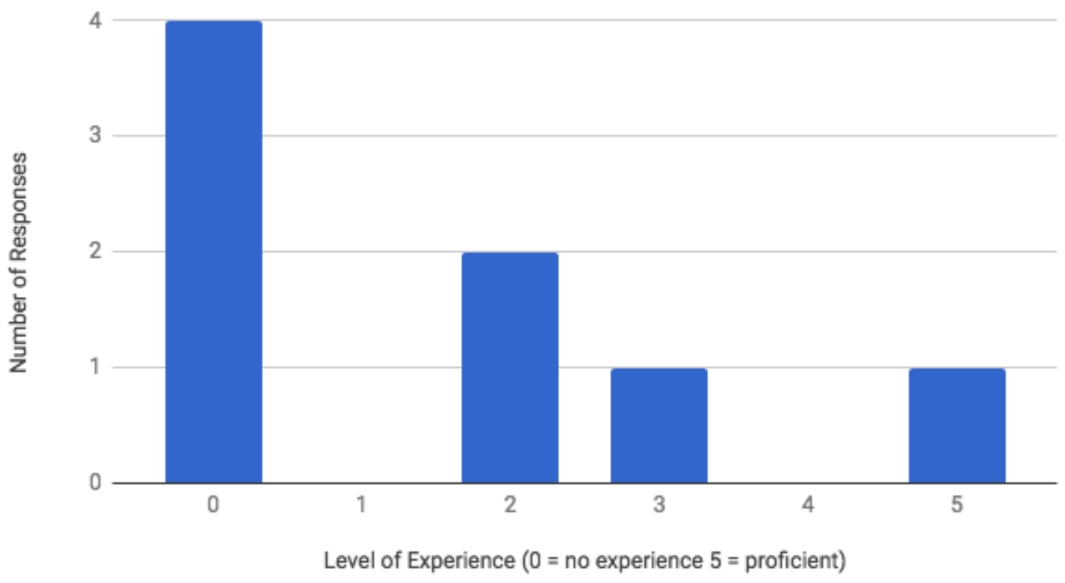


Figure 6.1: Usability Test Participant LabVIEW Experience: Amount of experience users has prior to the usability test. Half had at least one semester worth of experience, while the other half had none.

Table 6.2: Usability Test Participants Coding Experiences

Comfortable coding in programing language	LabVIEW	Other (HTML, C++, MatLab, JavaScript, etc.)	None
Number of affirmative responses (8 participants total)	4	8	0

During testing I asked and noted which programming languages participants were comfortable coding in. Two of the participants who reported that they had previous LabVIEW experience revealed to me during testing that they had not used the platform in at least two years, and encountered more errors than the other two users who stated that they used LabVIEW more recently.

For measuring the number of completed tasks and correct post-task assignment diagrams, if the participant completed the task, I note a 1 next to the Task number. If they did not, I note a number ending in 0.5 under “Task completed” indicates that the participant verbally or on paper noted that he/she understood the next task that followed the last task that he/she completed. For example, a 2.5 total denotes that the participant completed the first two tasks, and either explained aloud or wrote down the process for the third task, but did not complete this task. I also report the time it took users to complete each task, in minutes, based on screen recordings. If the user did not complete a task, but explained what he or she was trying to code conceptually, and therefore received a 0.5 in the “Task completed” column, I show the time it took the participant to work on the task plus his/her explanation time in the “Time to complete task" column. I note the number of correct post-task assignment diagrams as fraction of the number of correct diagrams over the number of attempts.

The Getting Started Utility connects the myRIO and ThingWorx accounts, but before each session, I deleted the preference files, so usability test participants had to log into the ThingWorx account and the enter the myRIO preferences with the login information I provided. All usability test participants used the same ThingWorx account credentials.

Participants tended to make several similar mistakes and errors, or encountered some of the same problems that let to obstacles in completing some of the tasks. I show a summary of the findings below for errors made by at least two participants. I considered an "error" to occur when a user asked me for help, made a coding or selection decision that would prevent his or her code from running properly, or made an attempt to produce code or features that LabVIEW or the Toolkit does not include. If the user ran a "successful" VI (LabVIEW did not give any errors), but failed to do so within the task's directions, this also counted as an error (for example, if the user ran code from the myRIO, but the task called for running on the laptop). I also note usability interface features that led to hindrances like accessing login information, which users should be able to access seamlessly to quickly initialize their accounts.

Table 6.3: Errors in LabVIEW

Error #	Description	# of participants who made error	Fraction of novice LabVIEW users	User quotes
1	Accessing a VI from a project	2	1/2	
2	Navigating to a block diagram from a front panel	5	4/5	
3	Correctly implementing the while loop	3	2/3	
4	Deleting code or broken wires	2	0/2	
5	Connecting a stop button to a myRIO button in a while loop	2	1/2	
6	Not knowing how to run a vi	3	2/3	[Novice user]: "Has that been my problem all along?! I had it this way at first, and then I was like 'why isn't it working?'"

Table 6.3: Errors in IoT Education Toolkit Feature

Number	Description	# of participants who made error	Fraction of novice LabVIEW users
1	Not finding the ThingWorx login from the Toolkit Express VI window	3	2/3
2	Failing to select a property on the Express VI window	5	1/5
3	Failing to select a Thing on the Express VI window	3	2/3
4	Using a front panel to create Toolkit controls	4	2/4
5	Using the wrong read or write block	5	3/5
6	Running the code from the wrong source (PC laptop vs. myRIO)	3	2/3
7	Not being able to navigate to the Express VI window after closing it	2	1/2
8	Wanting to make a cluster as an input/out for a write/read block or have write blocks accept more than one input	3	1/3

Other errors

One participant asked for help after becoming frustrated from trying to change property values from the Toolkit's Express VI windows, which this feature does not allow. It is unclear from screen recordings if other participants tried to manipulate the values here as well.

6.2 ThingWorx Dashboard

Like most cloud platforms, ThingWorx gives an online dashboard via web page that users can typically view and edit their cloud data. Students in the Fall 2017 robotics course, as well as those in the usability test, were only instructed to use the web page to view their data, but ThingWorx also has resources for users to be able to create gadgets, or property value representations in the dashboard, that allow users to not only view, but also control their data as an output. For the purposes of this beta release however, students were not expected to learn this function.

Pairing with the IoT Education Toolkit, the ThingWorx dashboard displays real-time data from all the properties of Things that users have created. In other words, as values being pushed to the cloud change, the properties displayed in the dashboard reflect this change concurrently. In the Getting Started Utility, when users are first creating a ThingWorx account and/or logging in for the first time, a window gives them a link to their dashboard where they can access this data (Figure 4.6: GSU, Adding a Dashboard). I will reveal the developers' original goals for how users interact with the dashboard and how students from the robotics course and usability test participants actually used it as well as their opinions on it. Below is an example of the ThingWorx page displaying data that I was writing to the ThingWorx cloud.

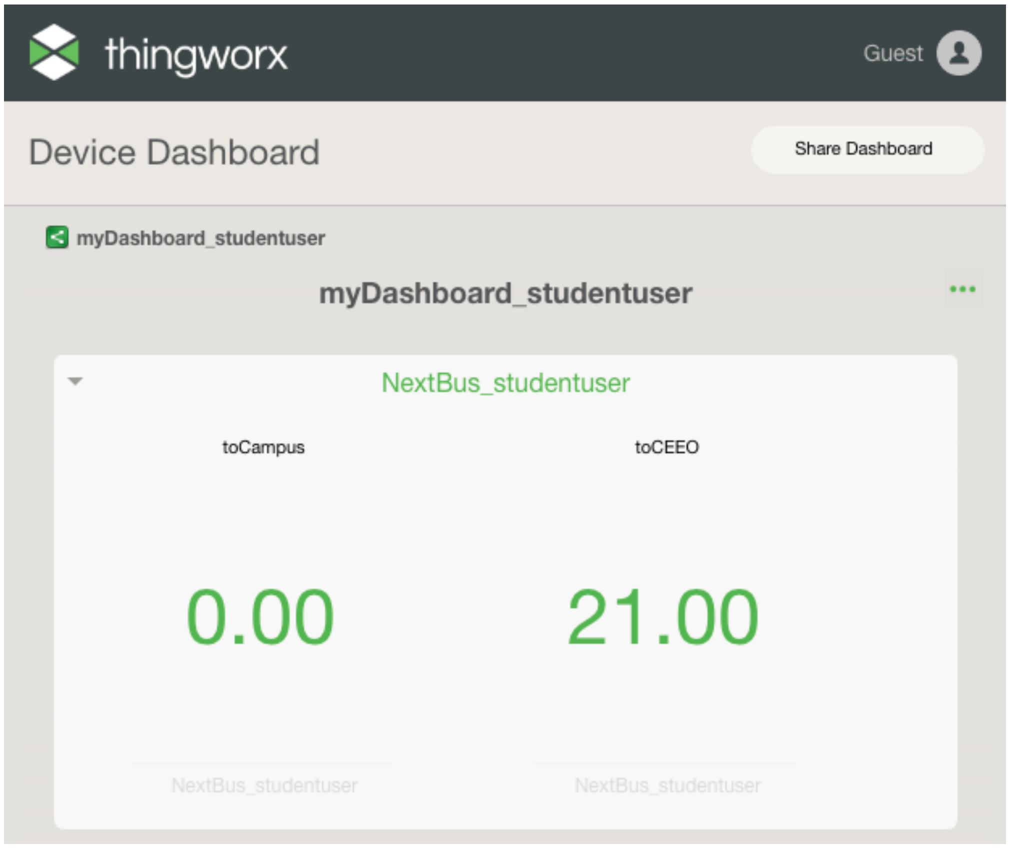


Figure 6.2: ThingWorx Dashboard: Values that are sent to or from ThingWorx. This is the dashboard for "studentuser." "toCampus" and "toCEEO" are two gadgets with number values under the "NextBus" Thing.

Here is an example of a dashboard displaying the data from the studentuser account. In this dashboard, there is one Thing (NextBus) displaying two properties in two different gadgets labeled "toCampus" and "toCEEO," representing the time, in minutes that two different buses are expected to arrive to different stops on the Tufts University Campus. I was writing this property from my laptop during the usability tests.

6.2.a Developer Interviews

An interview with the PTC developer revealed two different levels of goals that PTC had been aiming for in integrating the dashboard to the Toolkit.

Level 1

The PTC employee, whose job in developing the Toolkit was in part to create the dashboard so users could easily visualize the data, stated that at a minimum, the dashboard served as a "visualization that data was in [the cloud], and that it was changing." He goes on to say,

"Very often a student will create an ultrasonic sensor [for example] that is reading a distance, and they just want to see that. Seeing that data is essentially the gratification of 'I did something correct or successfully,' and that is very often associated with pretty much any new input or new information that comes in the platform. You always want to bring in that information, and then see it in someway."

This statement suggests that the dashboard can be a source of success, as well as a platform to debug. If data is not being sent to the cloud, it will not appear in the dashboard, resulting in an error. I will discuss other types of debugging methods, as well as that which uses the dashboard, in the "Debugging" section.

Level 2

The dashboard can be more than just a place to see data however, according to the PTC developer. He describes in the interview that he and his team were hoping for an additional layer of complexity of this feature to allow students to build on their IoT experience and knowledge. He states, that the hope here was...

“At the second level, the more complex level, that students would actually create some basic applications that would be meaningful, so that [data] would interact at a complex enough level so that they could control something, that they could actually operate a piece of equipment, or that would operate a more complex system that was potentially bringing in a lot of different inputs, and maybe averaging together or synthesizing that information in some way and then displaying the synthesis of that data instead of just raw data itself. That was the hope at that more advanced top-tier level.”

This developer outlines an ideal feature for the platform, from PTC’s point of view, which involves more data analytics and a stronger inclusion of complex algorithms for raw data.

6.2.b Student Interviews

The Getting Started Utility as well as the class instructor introduced the online dashboard to the students in the robotics course. During my interviews with these students, I asked them how they used the dashboard. If they did not remember what the ThingWorx dashboard was, I showed them a picture as an example. Some students mentioned the dashboard when answering other questions as well.

Table 6.4: Student Interview Dashboard Responses

Question	Response
How did you know when data was reaching the cloud?	"In a project, we would go to the task manager or the dashboard on the ThingWorx academic website, and then you could actually go into your Things and your properties and see if those values were changing." "If we were sure that the program [LabVIEW code] was set up to send data to ThingWorx correctly, then we would be checking [the ThingWorx dashboard] and see if there's actually that data value that's being sent."
How did you use the dashboard?	"I didn't ever use the dashboard" "The sense I got from the dashboard was that it's designed as a user interface to see what's going on very much for this sort of run-of-the-mill IoT Things of like, how many eggs in my freezer? What's the temperature outside? Are the window shades closed? It looked like it was built to take these values, sort of like the LabVIEW front panel. And I really wanted to look into see this JSON string. Does it have these values in it? and did we create a new JSON string each time, or is it overwriting? So, I think by the end of the class the level that we were using the ThingWorx IoT cloud for, the dashboard was not helpful." "I don't think I've ever seen that." "We would mostly use the dashboard just for cloud storage. We mostly just used it as a way to get data from point A to point B." "I understand what [dashboards] are, but I don't entirely understand their purpose when Composer exists. Unless you're controlling something from the dashboard, it's not useful."

Two of these five participants never used the dashboard, and one stating that he had never even seen it. The other three only used the dashboard on occasion, but understood the concept and correctly viewed it as a representation of their ThingWorx Cloud data, or a summary of ThingWorx Composer data.

6.2.c. User Testing

I introduced the ThingWorx online dashboard to participants in the pre-task video, gave them the link on their handout, and also had their assigned laptops opened to a browser page with the dashboard. When participants asked for my help, and I thought that they could use the dashboard to help them see if their data was flowing through the system correctly, I instructed them to try to debug on their own, to see if they would navigate to this page and use it effectively.

Below is a chart that illustrates the different use cases of participants interacting with the dashboard.

Table 6.5: Usability Test Participants Interactions with ThingWorx Dashboard

User	Interaction Description	Quote
LIAOH	Uses dashboard to debug. Tries sending a color value to the cloud (and Hue Lights Strip) during Task 1, but the lights remain the same color.	"So this ThingWorx page shows you like, what the current state is?”
BSKGF	Goes to dashboard web page to debug when she made some LabVIEW errors. Uses dashboard to see myRIO button value change during Task 2	“I really like [the dashboard]... It’s just really clear”
Y68JX	Thinking aloud before beginning task 1, after reading directions	“Ok, so this (the dashboard) must be feedback, so you don’t actually change [values] here”
Y5C8T	Looked at Next Bus values to understand Task 3	

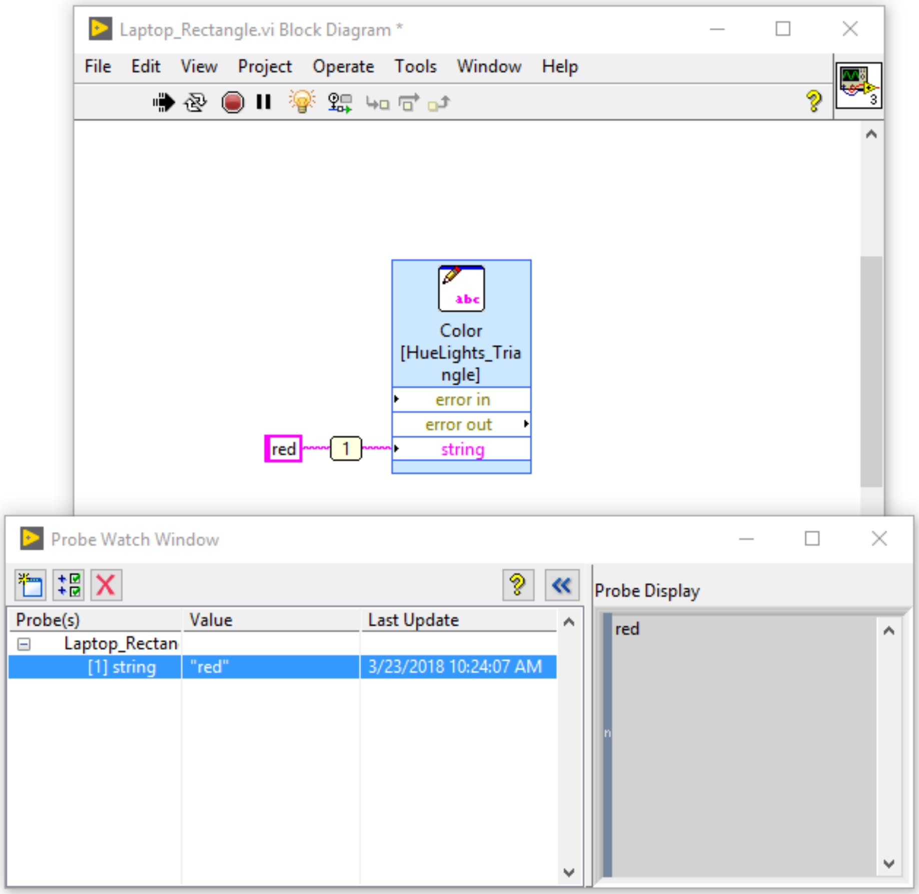
6.2.d. Summary

In general, students from the usability test had a more positive response to the online dashboard compared to the robotics course students. The usability test participants however were given fewer materials and resources, and some of them were novices to LabVIEW.

6.3 Debugging

As with most online systems, the IoT Education Toolkit can lead its users into making errors. Effective platforms will have means for users to debug in order to fix any errors or resolve any issues that may instead be a software, hardware, or connectivity issue.

In this section, I am considering "debugging" as the process of discovering why the system is failing to transfer the desired data properly. This includes for example, not changing the color of the Hue Lights Strip in Task 1 of the usability test, or not being able to run a LabVIEW program because of a coding error. The two areas where users can make mistakes here are either in LabVIEW (coding, myRIO connections, etc.), or in the IoT Toolkit features (sending or receiving data to or from the cloud). LabVIEW provides a way to debug this system, as shown in Figure 6.3 below.



Probe(s)	Value	Last Update
Laptop_Rectan		
[1] string	"red"	3/23/2018 10:24:07 AM

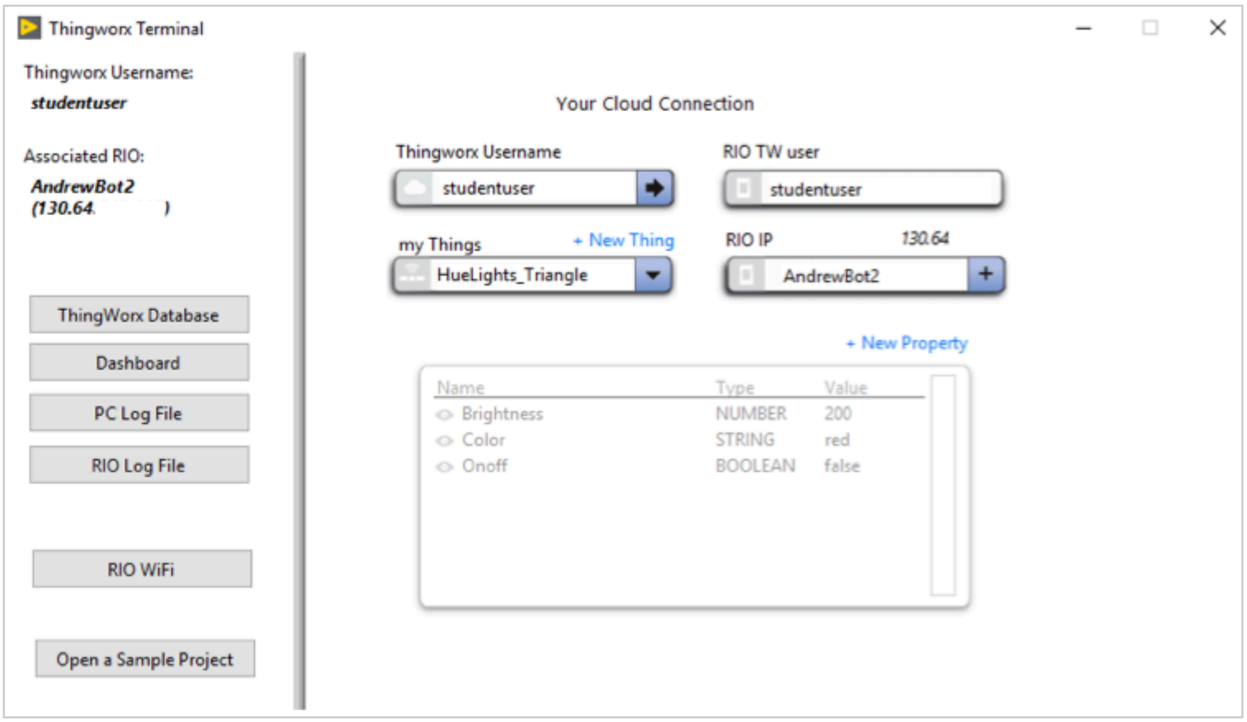
red

Figure 6.3: LabVIEW Probe: Users can press the light bulb icon in the LabVIEW toolbar and select a wire (denoted by [1] above) to see the values that are running through it. Helpful for debugging

After running code, users can press on the light bulb on the toolbar of the block diagram and probe the wire that is connected to a data source to see if it is receiving the desired input or output. Above, I have probed the wire connected to the write VI to see which value this VI is receiving.

This IoT platform also features several different methods to determine if data is flowing through the cloud properly.

1. The ThingWorx Dashboard. As I outlined in the previous section, the dashboard provides users with a source to refer to in order to view the data that they send to the ThingWorx cloud.
2. ThingWorx Composer. ThingWorx Composer allows users to view and edit their cloud data from an online webpage. The other two options do not allow users to write to properties. Similar to the dhasboard, this feature gives users another layer of reassurance that their data is indeed being sent from and/or to the cloud.
3. The IoT Education Toolkit Terminal. This element, shown below, can show users the values held in the cloud, as well as provide information about the associated myRIO and laptop.



Thingworx Username:
studentuser

Associated RIO:
AndrewBot2
(130.64.)

Thingworx Username: studentuser

RIO TW user: studentuser

my Things: HueLights_Triangle

RIO IP: 130.64

AndrewBot2

Name	Type	Value
Brightness	NUMBER	200
Color	STRING	red
Onoff	BOOLEAN	false

Figure 6.4: ThingWorx Terminal: Accessed through the LabVIEW toolbar, the terminal allows users to view and manipulate their ThingWorx data and myRIO system.

The terminal, shown in Figure 6.4 above, is a resource in the LabVIEW menu that Professor Rogers created for the students in the robotics class during the semester, but students reported that they did not use it, or used it only once. I also did not introduce this feature in the usability tests, so I do not include this in my analysis.

6.3.a. Developer Interviews

The NI developer stated during his interview that he hoped, "students would not have to debug at all," at least from the LabVIEW logistics side. The code behind the Express VI's should have been strong enough to allow students to run data successfully through the system and into the cloud.

From the ThingWorx side however, the PTC developer stated that the dashboard is a good source of measuring success, but not as powerful to exactly determine where an issue stems from. This cloud feedback is therefore limited, and he went on to talk about the complexities of having more than one microcontroller connected, and how that makes debugging more difficult. He stated…

"I think that's specifically the area that I think we could make it a little bit better because that comes down to how these Things are referenced and in more complex system interactions so right now the only way to really troubleshoot that is through ThingWorx Composer which has some challenges but if you're trained on it, it's definitely possible. So maybe bringing some of those strategies that exist in ThingWorx Composer down to the Toolkit level so that they're more accessible."

While Composer offers more insight into connectivity, hardware, and software problems, students would need more expertise in the platform to efficiently debug.

6.3.b. User Interviews

The students who I interviewed from the robotics class all had at least one semester's worth of LabVIEW experience when I interviewed them, and therefore I assume knew how to debug in LabVIEW using probing and accessing help menu's. I presumed these were the methods they were referring to when they said they tried "debugging in LabVIEW."

To ensure that data was reaching the cloud, two participants stated that they debugged by creating a "read vi" that was receiving property values from a "write vi" from the same code. Two different participants also stated that they used ThingWorx Composer to debug, and while all of them were exposed to the dashboard, none of them stated that they used it to debug.

6.3.c. User Testing

Only one participant, who had previous experience with LabVIEW, used the probing method to debug using LabVIEW. One other participant, who also had experience in LabVIEW, used the LabVIEW help menu to debug. The latter also tried to find a help menu for the Express VI's without LabVIEW, but this feature does not exist in the Toolkit.

Two participants, during user testing used the dashboard to debug, or receive the affirmative feedback that the PTC developer was referring to. Both of these users navigated to the dashboard after they tried to run their code to complete a task, but failed to create a successful code diagram to initialize the Hue Lights Strips, so the data never reached the cloud and therefore the ThingWorx dashboard.

Summary

LabVIEW and the IoT Education Toolkit offer different means to help make apparent to the user connectivity and programming issues. LabVIEW offers a way to see where the data is traveling through the code and what values are being transferred where, which students with LabVIEW experience typically use. A few usability test participants used the online dashboard to see their data reach the cloud, but this web page only affirms when a connection is being made successfully and gives no other descriptive feedback. ThingWorx Composer offers the best option for ThingWorx related problems, according to a ThingWorx developer, but this feature requires users to have more experience with the platform.

6.4 Adding Complexity

The IoT Education Toolkit, according to the NI developer, should be accessible to students without any IoT or LabVIEW experience. Eventually however, students who have not become comfortable with the system, will. In order to continue to allow students to expand on their learning of IoT and robotics, the system must have flexibility to allow students to create increasingly more complex systems.

There are different features, apparent and discreet, that allow students to go beyond simple activities like the ones I present in the usability tests. The myRIO for example, is a highly-equipped hardware device that would allow students to develop more sophisticated systems while using the same equipment they were presented with on the first day of class. Adding data analytics and complex decision-making algorithms to IoT solutions is another layer that students could add to their solutions. Although the current version of the Toolkit does not include data analytics, I will make an argument that it should be a feature of the platform since it is an important part of IoT.

6.4.a Developer Interviews

PTC

The developer from PTC stated that he hoped students would eventually try to create more complex IoT systems and discover the opportunities in ThingWorx Composer that would allow users to develop machine-learning algorithms and data analytics. I summarize the interview participants' responses to the questions below. You can find the full transcript to the responses in (Appendix E).

Question: *Did you see any clear limitations on the Toolkit as they relate to allowing students to add complexity?*

"The cloud aspect is difficult... largely because there's no standard... There's no good way to provide a student with a way to generically add new sources of data because there's no shortage of those. There's hundreds of different sources of data. There's baseball data, there's weather data, there's geological data. So all of those exist, but there's no standard interface yet for those Things to be pulled in..."

“That concept extends to Things like analytics and to Things like machine learning. Some of those more advanced concepts, because there's no way to directly pull some of those engines and tools in to work on your data, because there's no standard interface for those either.”

He continues on to mention that ThingWorx is a powerful platform that can integrate these large amounts of data, but, “to open up the complexity of that, you just have to kind of be a fully trained software engineer,” which is outside the scope of an undergraduate classroom, unless a student already has experience with the platform.

Question: *How did you hope that students would use Composer, if at all?*

Elaborating on the question I had previously asked, the developer continued to explain the features of ThingWorx Composer and states, “the hope was that a student would eventually get to that point,” if they came across a specific use case where they wanted to develop more advanced analytics or send data to another platform.

He further explains, “That was a hope. I'm not sure how many students have gone from a simple use case that they were using the Toolkit for, and then extended on to that next step, but where [ThingWorx Composer] fit was definitely as a more advanced and intricate version of the cloud aspects of the Toolkit.”

The most powerful IoT systems contain some kind data analytics or intricate levels of data sharing, and the Toolkit’s partners from PTC, a leader in the IoT development industry, tried to integrate, or hoped there would be some integration, of these features.

National Instruments

As part of the collaboration with Tufts University, NI allocated myRIO's to the University partners, so students in the robotics class had this resource throughout the semester. The NI developer who I interviewed argued that the myRIO is an acceptable piece of hardware to pair with the Toolkit, as the platform allows users to code in LabVIEW in order reach or send distinct property values of the microcontroller. However, the hardware is expensive, and is "overkill" for many simple IoT solutions. I summarize some of his responses to interview questions relating to adding complexity below. You can find a full transcript of these responses in Appendix E.

Question: *Do you see any clear limitations on the Toolkit as they relate to allowing students add complexity?*

"Certainly being able to take the myRIO I/O and customize what parts of that you want to write to the cloud to write to ThingWorx and vice-versa. If there is some data that's in the cloud that you want to be able to reflect on this edge node, the myRIO, the IoT Toolkit is a good way to do it. You need some code running on that target and there might be a better way to program that in the future but today the only way to do that is to either write a pre-configured image that's reading and writing all of the data and sort of to deal with the latency there or to write some custom code using this Toolkit to deploy it from LabVIEW onto the myRIO and now it's capable of talking to these cloud services all right you're saying this is."

"...If the goal is to make a myRIO a cloud connected device and then you need something like this. If the goal is to inexpensively measure temperature of homes, [it's] a terrible idea."

He argues that the myRIO offers useful customization and adaptability with LabVIEW, and in making it an IoT connected device, the Toolkit offers a successful framework to do so. In the following response, he does state that this hardware offers more features than students need to create simple IoT solutions.

Question: *How did you think the myRIO fits as a microcontroller that pairs with the Toolkit?*

"For these types of projects, when the end goal is to read on an analogue measurement and you're content with ten bits or twelve bits of accuracy on an analog sensor reading or you need to blink an LED [for example], you don't need any of the power of an FPGA processor, 32 bits with 512 mega RAM and all of this other horsepower and all the protection of the I/O and all the other Things that the myRIO brings to the table for student applications, at $2.50 Arduino has more than enough horsepower for those applications and some of these platforms do have Wi-Fi connectivity and have the ability to do those Things…."

"...And so you know if you're competing against [Arduino's straightforward ecosystem], and [in this IoT Toolkit] you're bringing this sort of heavyweight solution in order to Blink an LED, you're bringing the wrong set of equipment to the fight..."

The myRIO, he suggests here, cannot be compared to other simpler microcontrollers and platforms like the Arduino, as they offer different frameworks and goals for connecting different types of data, but the myRIO is still too complex and expensive for students to appreciate or utilize all its available features.

6.4.b. Student Interviews

Below is a chart displaying the common responses or feedback to features of the Toolkit. You can see the full transcripts for selected responses in Appendix F.

Table 6.6: Student Interview myRIO/ThingWorx Adding Complexity

Feature	Response/Feedback	Number of Participants in Agreement (5 total)
myRIO	Unnecessarily “robust” or “powerful”	3
	Issues with initialization (connecting to Wi-Fi, or loading software)	3
	FPGA system is useful	3
	“User friendly” or “easy to understand”	2
Toolkit/ThingWorx	Wanted to explore, or was curious about other IoT microcontrollers pairing with the system	3
	Time delay was an obstacle in creating successful robotics projects.	2

6.4.c. User Testing

During the usability tests, I asked participants to either run their code from the Laptop (PC) or myRIO. In the pre-task video, I explained what the myRIO is and some of its features like the accelerometer and FPGA system, but did not ask them to integrate any its high-level features as part of the tasks.

I asked the participants, as part of the post-task survey, I asked the participants their opinions on the myRIO. The responses are shown below, as well as how much prior LabVIEW experience the participant reported to have.

Table 6.7: Usability Test Participant myRIO Opinions

Response to Survey Question: *Please explain how well do you think the myRIO microcontroller pairs with the Toolkit.*

User	Response	LabVIEW Experience (0-5, 0 = no experience, 5=proficient)
UP1	I'm sure it's more useful for other, more complex systems. Plus using LabVIEW with myRIO with easy	2
UP2	I think it pairs well and makes it clearer	0
UP3	It worked but the LabVIEW UI is so frustrating to use	2
UP4	They seem to pair well, there is a bit of delay but it's fine	0
UP5	Apart from connection, not many problems	2
UP6	It's a good black box tool for general understanding	0
UP7	Seemed to work well	0
UP8	I feel like this study did not show me everything the myRIO can do and I don't have enough information to answer this question yet	5

In the "Additional Comments" section of the post-test survey, participant UP1 stated, "Since you run this off your computer without the myRIO, I'd be interested to see how myRIO can enhance IoT systems (like what can it do that a laptop can't?)..."

In general, participants did not encounter many issues from the myRIO. Two of them reported having connection or delay errors, and two noted that they were aware that there are additional capabilities of this hardware, even though the tasks did not ask them to uncover all of the features.

6.4.d. Summary

Overall, participants generally understood that the myRIO may be more complex and expensive than undergraduate students would need. Student interviews revealed that they did however appreciate the FPGA system when building their projects in the robotics class, but these students as well as the usability participants reported some delays or obstacles in initializing the myRIO.

6.5 Sharing Data

IoT platforms offer different methods to exchanging data values or projects with other users in the community or with other hardware devices or Things. While undergraduate students may document their projects to build a personal portfolio for developing their professional profiles, their primary focus is sharing projects with professors and teaching assistants in class.

From a different perspective, sharing data also involves communicating within the established IoT network. This could include Things “talking” to each over the cloud and making decisions based on this interaction. Exchanging information between users also involves security concerns however, and developers must consider privacy and limitations to sharing data.

I will focus this section mainly on how students worked together to share data to develop projects in the robotics class. I will briefly discuss security in the system, but this was not a crucial concern for the class and for developers at the beta release stage, since the students would not be exchanging sensitive or personal data. They all used the same ThingWorx account, including the login credentials.

6.5.a Developer Interviews

The NI developer, in response to my question of “How did you expect students to share their data with one another?” said,

“It seemed like there was a push to be fairly collaborative in the shared use of data, but I didn't know what that looked like and I didn’t know whether that was going to be really important to the Toolkit. It seemed like it ended up being fairly important to the robotics class’s use of the API because everybody was creating a piece to larger whole, and so everything- you need to be able to look at that data.”

Even though he was not entirely involved in the development of privacy settings, the NI developer understood the need to allow users to exchange information and data to each other in order to successfully explore the Toolkit’s features.

The PTC developer was more involved in creating the privacy settings featured in the Express VI window, show in in Figure 6.5 below.

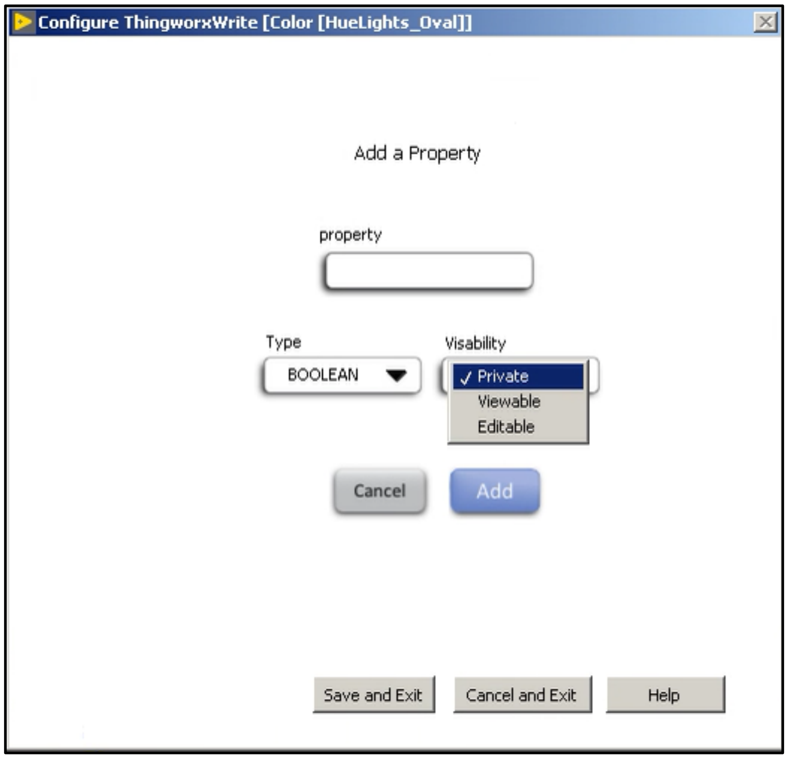


Figure 6.5: ThingWorx Add Property Privacy Settings: Users can adjust the visibility settings so other ThingWorx users can or cannot view or edit the property.

These three settings allow users to either make their values editable, so anyone with access to a ThingWorx account can manipulate them from ThingWorx Composer. The privacy setting to a new Thing is similar to this structure.

During a discussion about security settings, the PTC developer explained, “In an educational context, [security] is always kind of pushed to the side, and it's for good reason. Almost always that information is not valuable or vulnerable to threat, so you don't necessarily have those [security] conditions ... It's definitely not the priority I don't think but you always have situations where you don't want a malicious user to take down all access to this learning opportunities.”

He justifies the lack of security around the current system by realizing that the shared information is not sensitive.

6.5.b. Student Interviews

Students in the robotics class all submitted their projects online via a Tufts University portal. Student interviews revealed that when collaborating outside the class, they would use Facebook or email to communicate with one another when they were having trouble with the platform.

During a conversation I was having with one interview participant (RS1), I asked her where she thought the Toolkit was useful. She replies,

"Anything where multiple people are trying to talk to one Thing, or you have multiple robots that you're trying to send to a central location. Allowing devices to communicate with each other is the biggest Things. Maybe that's a Mac to a PC device, or that's many robots a computer, or computer to many robots, makes it a lot easier."

She explains that the Toolkit made it easy for her to connect Mac computers with PC's, as the Express VI's could easily pull or send information from both sources. The Toolkit also serves as a gateway to talk to multiple devices from different sources, which this participant found useful.

During another conversation with another member of the class (RS2) he shared with me a problem that he and his classmates were having with deleting properties. He said, “I remember there was some confusion with the UI related to- We create properties for Things and they’d get reset whenever they’re edited, so with the final project, there are quite a few messages, emails sent to the class saying ‘whoever deleted everything, send me a Facebook message right now!’”

In this case, having the platform being open for all students in the class created a temporary problem for students. He goes on to explain to me that eventually, everything was fixed.

6.5.c. Summary

Sharing data between users can be difficult for IoT developers, but luckily in academia, professors who stress project-based learning often require and encourage exchanging information and data between students. The robotics class at hand shared data to create complex and intricate IoT solutions in the classroom.

Chapter 7 - Analysis

The results from the interviews and usability tests indicate that this IoT platform has some obvious areas of improvement or additional considerations if developers wish to create another iteration of this Toolkit. NI and PTC collaborators could close many of these gaps by adding some additional features or descriptions, changing some of the user interface designs, and developing a descriptive guide or help menu that pairs with the Toolkit to help students navigate through and leverage all its features. I organize this section into the five measurable areas of the Toolkit, as previously discussed. I will first discuss the user's (robotics students and usability test participants) common errors, misunderstands, or mistaken expectations, and then classify each one as an identified gap and make suggestions for improvements.

7.1 The Getting Started Process

Although the robotics class students were nearly six months removed from the start of their experience with the Toolkit by the time I interviewed them, most of them did recall going through the Getting Started Utility. This feature was originally meant for students to initialize their myRIO's with the correct software and connect to their assigned ThingWorx account. As part of the getting started process, I also studied how novice users interacted with the platform for the first time.

7.1.a. Discussion of Findings: Student Interviews

I outline student expectations and understandings of the getting started process of the Toolkit as follows. I derive these points from student interviews:

1. Download process of software or drivers

Past experiences with LabVIEW and myRIO installations caused students in the robotics class to expect a Getting Started Utility that installed appropriate software to the machines. Typically, they only need to undergo this process once, but they found that they often had to reinstall the software when they changed partners and therefore had to pair a new myRIO to their computer.

Gaps

While the development team clearly met the expectation of including downloads in the Getting Started Utility, they failed to meet the students' expectations of incorporating these initializations smoothly enough for users to avoid any frustrations. This then reveals a gap between *the Expected and Perceived Service from the Users Perspective*. While students in the robotics class may have understood this message (after some possible help from the professor and/or other students in the class), it still raised irritations and hindrances, especially when the student began working with new partners.

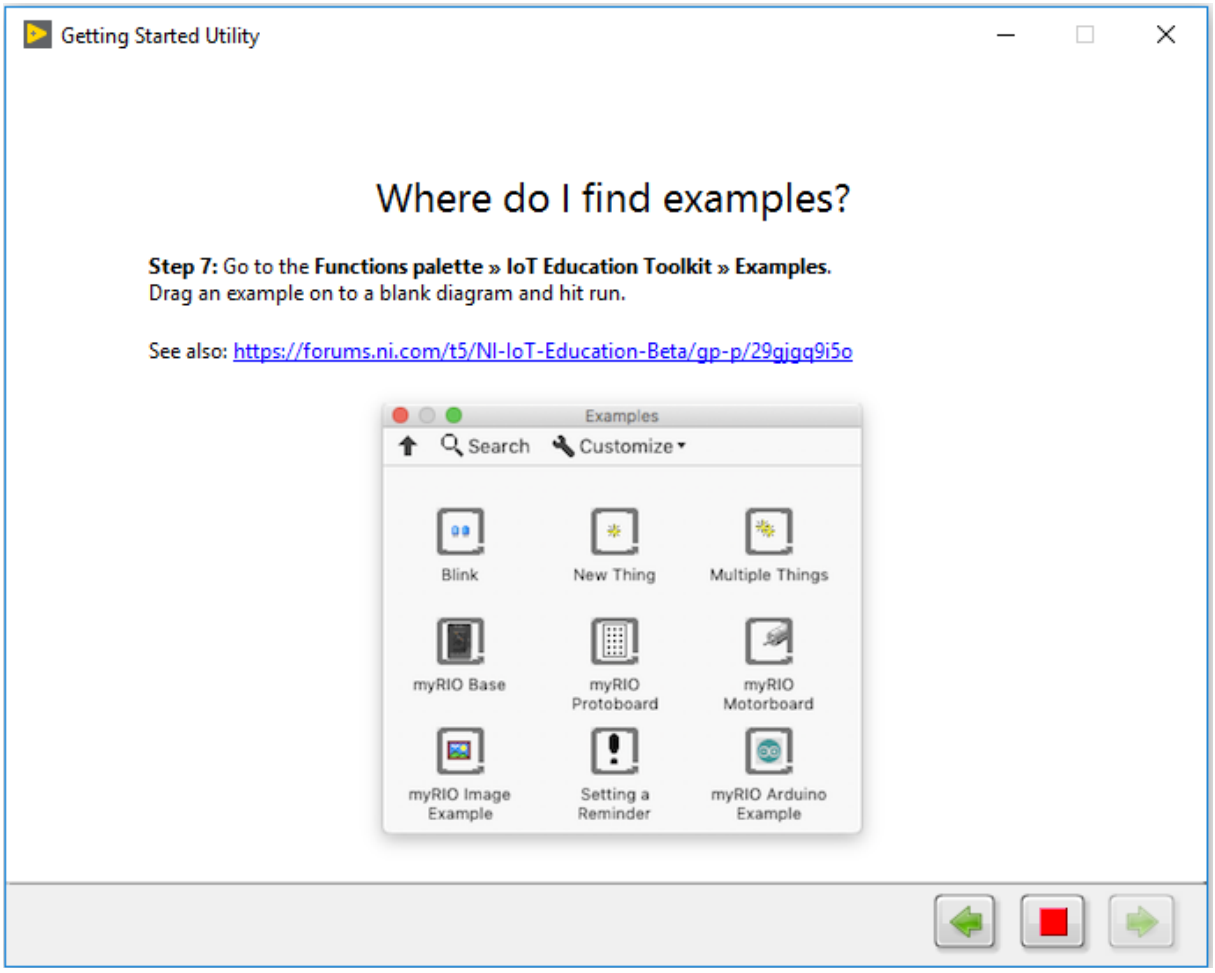
Improvements

To address this gap, developers could include a more effective description to outline what this error is, how to fix it, and note which version of the software the machines are running (and from which username). A link or outside source to a video or longer description should be available in the user interface for students to grasp a better understanding of the concept, but may not be appropriate in the Getting Started Utility if it is essential to copy preferences or load a new image every time a myRIO is paired with a different computer. Having a long description in an initialization like this will be redundant, and since students have admitted to skipping over these descriptions throughout the Getting Started Utility, many users may ignore a longer description.

2. Examples or demonstrations (either in LabVIEW or online)

Some students also expected that the Getting Started Utility would be paired with some examples or demonstrations of how the VI's could work to send or receive information to/from the ThingWorx cloud.

Gaps


Getting Started Utility

Where do I find examples?

Step 7: Go to the **Functions palette » IoT Education Toolkit » Examples.**
Drag an example on to a blank diagram and hit run.

See also: https://forums.ni.com/t5/NI-IoT-Education-Beta/gp-p/29gjgq9i5o

Figure 7.1: GSU Finding Examples: Users can access information on where to find examples in the last step of the Getting Started Utility

The Getting Started Utility actually featured a page in the last step of the Utility that suggested where to find examples (see Figure 7.2 above), but students did not address this, and possibly found them as too large of a leap from the Getting Started Utility to creating their own code. Having a clear coding example within the Getting Started Utility could bridge this gap that lies between *the Developer's Translation of Perceptions of Consumer Expectations into Service Specifications and the User's Perceived Service.*

Designers of the Toolkit made an effort to include examples, but the interviews revealed that not all users saw this feature.

Conversely, because the students failed to report this example, *their perceptions of the Toolkit were detached from the actual Toolkit* as a result of the gap mentioned above.

Improvements

To address this gap, developers should short example code diagrams in the form of a "dummy" screen, an interactive VI (one that students can change and expand on themselves), or a short video that students can watch quickly to grasp a quick understanding of the Toolkit's features. Having a clearer interface in the GSU could also promote users into reaching the last page of the Utility through a progress bar, for example.

3. Fast process that would succinctly display or outline what the Getting Start Utility was doing

Some user interview participants reported that they like to work quickly, and ignored the graphics and descriptions featured in the Utility, and others stated they failed to understand what was going on during this process.

Gaps

Users often ignored the descriptions or written instructions that the developers provide to outline the flow of data in the Getting Started Utility, revealing a gap between the *Expected Service and Perceived Service* of the user.

Although developers make an effort to outline the cloud's data flow effectively, students still often misunderstood the concepts in the Getting Started Utility including how data is flowing from the laptop, through the cloud, and onto the myRIO. Developers realized this however before the beta release and did not make any adjustments to the system, nevertheless leaving this area with a gap between the *Developers' Translation of Perceptions of Consumer Expectations into Service Specifications.* Because the developers realized that the students would fail to benefit from the graphics, but continued to use them anyway, a gap lies in the translation from this understanding from the developer's point of view, and the implementation of their realizations into service specifications for the system. Developers failed to change this design of the Utility due to a time constraint, and they also recognized that during the robotics course, the students would have help readily available from their professor, who helped create the Toolkit.

Improvements

While the GSU stands as a necessary feature for software installments and logins, NI developers later decided that consumers would ignore the displayed LabVIEW code, or confuse them, and therefore would not be an appropriate element of the utility. A more effective approach to this Getting Started Utility that would diminish both the gaps mentioned, is implementing screens that feature only the necessary information to create the credentials needed for the computer and myRIO initializations. This method may exclude any LabVIEW teaching tools or information on how it is working, but allow the user to go through this process as quickly as possible without becoming confused or frustrated. Developers could however implement non-intrusive elements to implement in the program through small graphics that offer an opportunity for the user to navigate to more information about how the code is running and how the data is flowing through the system.

4. Clear display if an error occurs and directions to debug and/or fix this error

In addition to clear descriptions, students expected real-time feedback during installations and initializations. For example, when a user goes through all of the steps in the Utility successfully up until the myRIO real-time display (see Figure 4.9: GSU, Connecting myRIO & Cloud). If the device is not initialized properly, the data on the accelerometer graph will not change even if the user moves the device. The onboard LEDs will also fail to turn on/off at the user's command, as they should.

Gaps

In addition to clearer descriptions, students expected real-time feedback during installations and initializations. They also expected to be able to delete a Thing or property from the Express VI window after they created one by accident. This lack of feedback and navigation abilities creates a gap that separates the *User's Expected Service and the Actual Service*, since the specifications of the Toolkit do not offer these features.

Improvements

To resolve this gap, designers could create pop-ups or icons featuring informative information to notify the user of the error. For example, as one robotics student describes, a user may think he is pushing to the cloud from the myRIO, when he is not because the feedback from the GSU allows a user to try to initiate the LED's on the hardware as well as show live "dummy" values on the accelerometer graph (Figure), even though the myRIO is not responding to the LED commands or pushing information to the graph. Having graphics to describe this error would help students correctly system the myRIO.

7.1.c. Discussion of Findings: Usability Test Participants

One of the major hindrances of the usability tests emerged when users had no experience with the LabVIEW coding platform. The four users who reported that they had prior LabVIEW experience created post-task assignment diagrams correctly on 13 of the 15 total attempted diagrams (86.67% correct), while the four users who had reported no prior LabVIEW experience created 3 successful out of 11 attempted diagrams (27.27% correct) (Table D.9). These findings suggest that in order for novice users to understand the concepts behind IoT while using this platform, it is crucial to have at least some LabVIEW experience or more coaching than the information I provided. Only one of the participants referred back to the online video I provided for them. This user re-watched part of the video to discover how to create a constant on a LabVIEW vi. The handout I gave to participants (Appendix C) showed them the following

- How to reach the Toolkit Express VI's
- How to create constants/controls on the Express VI's
- How to create a while loop (including how to create a stop button on the front panel)
- Where to find the myRIO button block in LabVIEW

Even with the handout in front of them, participants sometimes asked me for help before I referred them to the handout again, suggesting that some tasks could have taken participants less time had they read and understood the “Helpful Tips” handout completely.

Below is a list of other usability errors that hindered participants’ abilities to complete the tasks without errors, as outlined in the Results section and Appendix D. These errors either stemmed from an error in the LabVIEW platform, or from features in the Toolkit itself. Errors and hindrances in LabVIEW range from accessing a VI from a project, navigating to a block diagram from a front panel, correctly implementing the while loop, deleting code or broken wires, connecting a stop button to a myRIO button in a while loop, and not knowing how to run a vi.

The actions that caused confusions or errors within the Toolkit itself include not finding the ThingWorx login from the Toolkit Express VI window, failing to select a property on the Express VI window, failing to select a Thing on the Express VI window, using a front panel to create Toolkit controls, using the wrong read or write block, running the code from the wrong source (PC laptop vs. myRIO), not being able to navigate to the Express VI window after closing it, and wanting to make a cluster as an input/out for a write/read block or have write blocks accept more than one input.

LabVIEW

I classify the following errors as “LabVIEW” errors nice they do not necessarily involve the IoT Toolkit. In other words, if I instead asked these same participants to perform a different set of tasks strictly in LabVIEW without involvement of the IoT Toolkit, they would have conceivably make these same mistakes or have been confused about how to access the same features I outline in this section.

I am classifying all of these errors as a gap between the *developer’s perceptions of consumer expectations and the consumer’s perceived service*. Developers assumed that LabVIEW would be simple for novice Toolkit users to use, especially if they already had coding experience like these usability participants had. Because I have made this categorization of all LabVIEW errors, I will only discuss the errors themselves.

Since users can avoid making these errors through having more LabVIEW experience, I do not designate sections to outline future improvements that can help make LabVIEW easier to use. Developers should however consider pairing the Toolkit with a help guide that features all of the errors I outline in this section.

1. Accessing a VI from a project

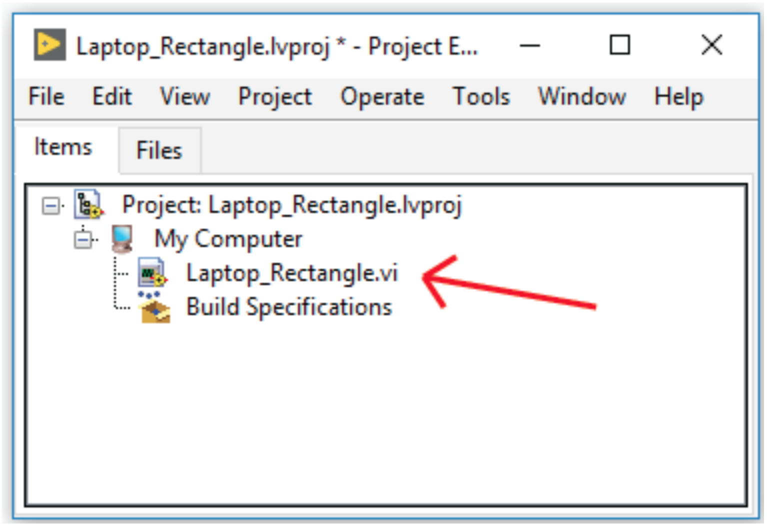


Figure 7.2: Opening a VI: Example of a project ("Laptop_Rectangle") that is running from a computer with VI "Lap_Rectangle." The red arrow is pointing to the VI

Half of the participants did not know how to open a VI with the project open in front of them. Two of these four participants claimed they had LabVIEW experience but had not used the platform in at least two years. To open up the VI that I provided for them, users would have to select the "Laptop_Rectangle.vi" from the project shown above. I did, however, try to guide users to this VI in the video. Below I show the Laptop project with circled figures to show users where they could find these VI's. I also show the myRIO project and VI similarly.

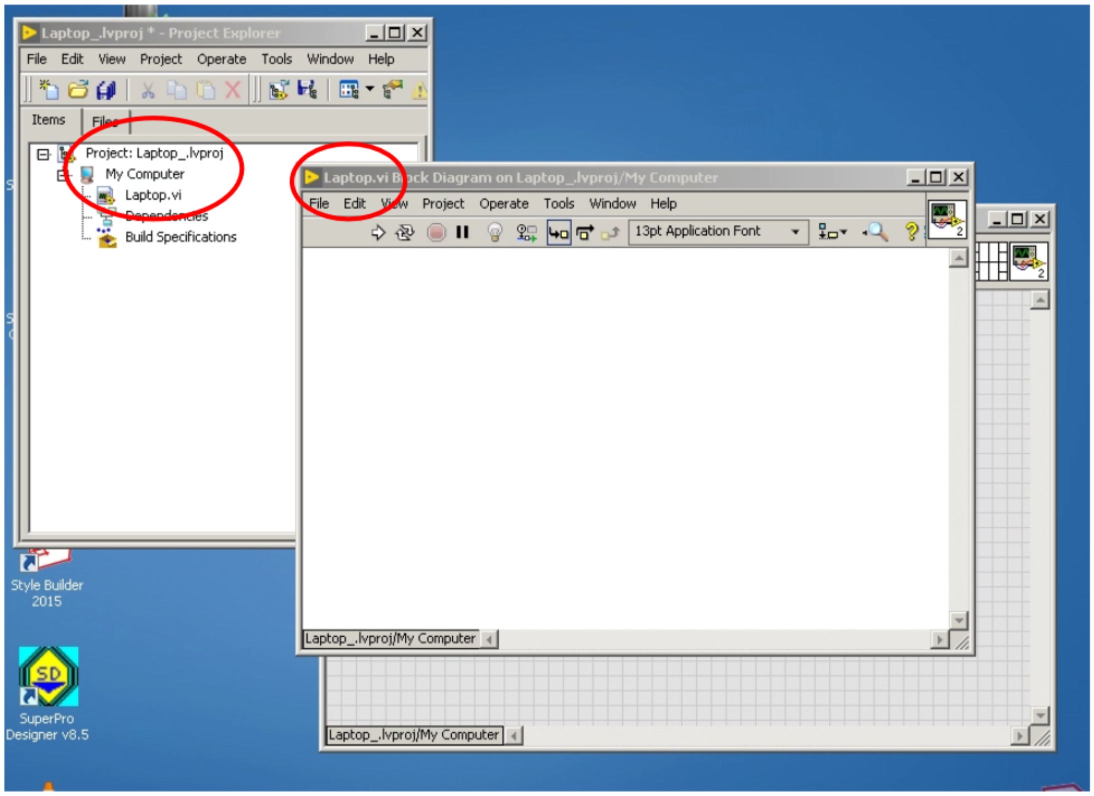

Laptop_.lvproj * - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files
Project: Laptop_.lvproj
My Computer
Laptop.vi
Dependencies
Build Specifications
Laptop.vi Block Diagram on Laptop_.lvproj/My Computer
File Edit View Project Operate Tools Window Help
13pt Application Font
Laptop_.lvproj/My Computer
Laptop_.lvproj/My Computer
Style Builder 2015
SuperPro Designer v8.5

Figure 7.3: Pre-Task Video, Where to Open a VI: During the video, this image was featured to show users how to open their VI's (right) from their project (left)

Even with my efforts to direct users to the correct files, they still had trouble opening a VI from a project. Developers therefore need to consider additional scaffolding for novice users to the platform and LabVIEW that involves this simple task.

2. Navigating to a block diagram from a front panel

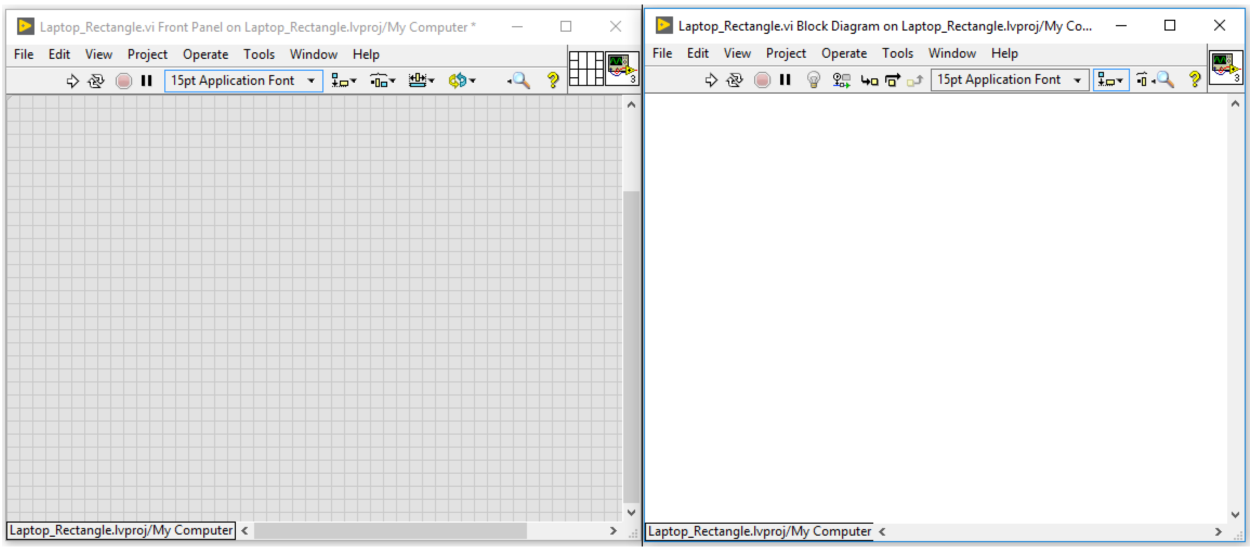

Laptop_Rectangle.vi Front Panel on Laptop_Rectangle.lvproj/My Computer *
File Edit View Project Operate Tools Window Help
15pt Application Font
Laptop_Rectangle.lvproj/My Computer
Laptop_Rectangle.vi Block Diagram on Laptop_Rectangle.lvproj/My Co...
File Edit View Project Operate Tools Window Help
15pt Application Font
Laptop_Rectangle.lvproj/My Computer

Figure 7.4: LabVIEW Front Panel & Block Diagram: Two parts of the LabVIEW development scheme. The Block diagram (right) is where users can create coding logic and initialize Express VI's. The Front panel (left) allows users to control values dynamically while the code is running.

Above are diagrams from a Laptop VI. The image on the left shows the Front Panel, where LabVIEW users can manipulate controls and view indicators. The Block Diagram on the right is where users create and edit block code. When users first open a VI from a project, the Front Panel appears, and they must navigate to the Block Diagram to code. Five participants made errors trying to navigate to this Block Diagram. All four novice LabVIEW users made this error.

Without proper coaching, this process is not intuitive, and I failed to give proper coaching to the participants to being coding here. This lack of knowledge lead to several participants also making the IoT Education Toolkit error of trying to create code on the Front Panel, which I outline later in this section. To avoid this error, developers should also include this feature in a compiled help menu or guide for novice LabVIEW users wishing to access the Toolkit, or point them to an online resource.

3. Creating a successful while loop

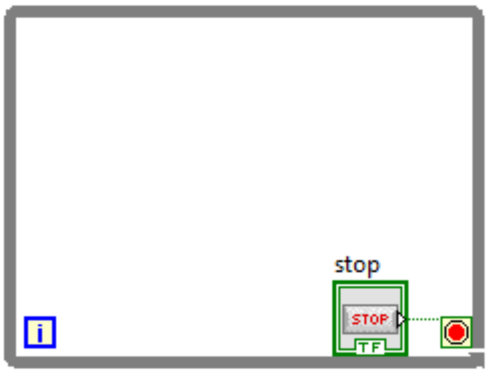


Figure 7.5: LabVIEW While Loop: Components of a while loop in LabVIEW necessary for a successful diagram. A user must connect a stop control or constant

Figure 7.6 above is an example of what I am referring to as a "successful" while loop. I outline this process in the "Helpful Tips" handout, but three of the eight participants still failed to create a while loop as such without help. The participants needed assistance in creating a stop button control or constant, which LabVIEW requires in order to be able to run the code. One participant also needed help realizing that she needed to expand the diagram in order to fit her code inside. Two of the three participants who made this error were novice LabVIEW users, suggesting that participants definitely need proper coaching before creating this loop, especially since the given directions were not enough.

4. Deleting code or broken wires

Two participants, both who had prior experience with LabVIEW, had difficulty deleting code from their Block Diagrams during the task. These participants stated that they had not used LabVIEW for at least two years. One participant tried to select the code and press the “delete” button from her laptop, but could not do so since she was still running code (see Figure 7.7 below).

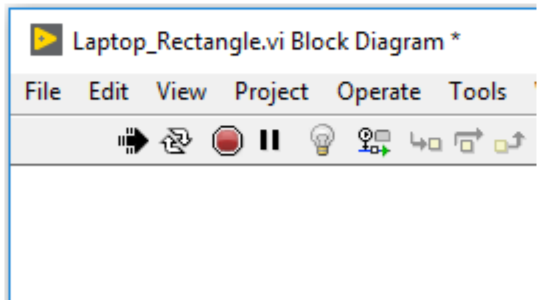


Figure 7.6: LabVIEW Code Running: Image of LabVIEW code running a VI successfully. The white arrow changes to black and the stop button turns bright red to allow users to end the program from this task bar

The diagram above shows code that is running in LabVIEW, indicated by the arrow that has changed from white to black. This participant also had previously failed to remember how to run LabVIEW in previous tasks, suggesting that the run button failed to attain significance to her. LabVIEW’s lack of feedback hindered this user from navigating through the platform efficiently in this case.

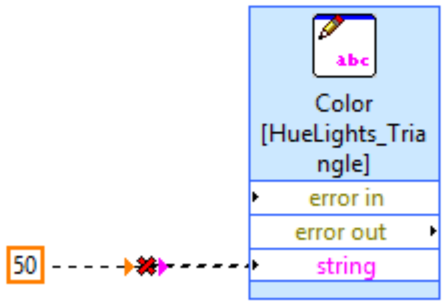


Figure 7.7: LabVIEW Broken Wire: Broken wire in LabVIEW denoting a mismatch in value types. A number value cannot control a string.

Figure 7.8 above shows a broken wire, which is LabVIEW's feedback to the user to tell him or her that there is an input/output type mismatch. Here, a number output constant is trying to connect to a string input. One of the users encountered this error and made frequent errors in trying to delete the wire. LabVIEW offers shorthand help to delete broken wires like this, but this feature is not intuitive and should also be including for LabVIEW user help that pairs with the Toolkit.

5. Connecting a stop button to a myRIO button in a while loop

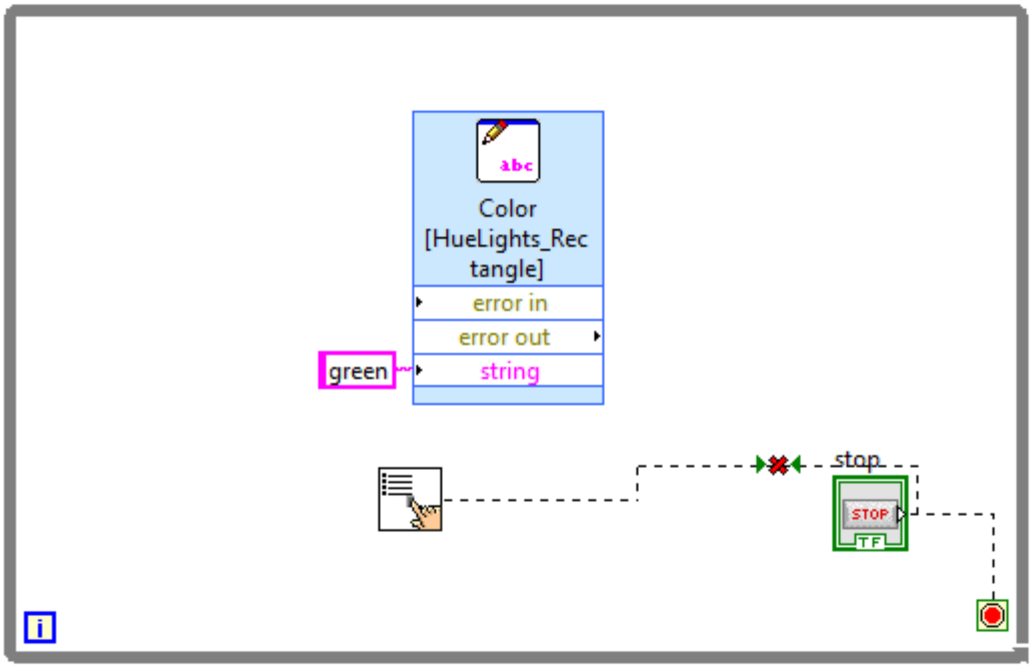


Figure 7.8: MyRIO Button to While Loop Stop Button: Example of an error users made in testing during Task 2. They tried shutting off the Hue Lights Values by ending the while loop and program by connecting the myRIO button VI (white block with hand pressing a button) to the while loop stop button.

The above figure shows an error that two participants made while they were trying to complete Task 2, which asked them to create code that caused the button on the myRIO device to change the state (on/off) of the Hue Lights. I showed users how to find the button VI, which reads the myRIO button value and creates a boolean (true/false) output from the VI with the hand pressing a button. In the first task, users created the Express VI with a color constant, which successfully changed the color of the Hue Lights Strip, without the while loop.

Unpacking this code, a user could understand this as “when button is pressed, stop the while loop.” They may have figured that stopping the while loop would end the Express VI program and therefore shut the Hue Lights Strip off. This concept is incorrect however, since the code is not sending an on/off value to the Hue Lights state, but to the while loop. To complete this task correctly, the users needed to change the Express VI to read the property “Onoff” in their Hue Lights Thing, and then connect the myRIO button boolean value to the input on this Express VI. With coaching, these two users seemed to understand the concept, but this further demonstrates the need for proper LabVIEW instructions and feedback before beginning to use this IoT Education Toolkit platform.

6. Not knowing how to run a VI

Figure 7.10 below shows the taskbar of a LabVIEW VI. The white arrow represents the run button to initiate the written code.

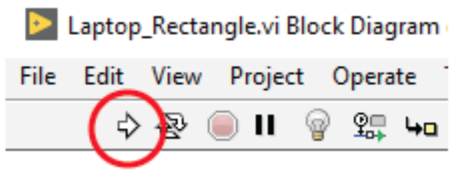


Figure 7.9: LabVIEW Task Bar: Selecting the white arrow in the task bar runs the code for the associated VI

Three of the users in the usability test could not deduce that this white arrows is the run button and either asked for my help during the test, or received my help without asked after they were looking for visual feedback without running the code properly. Two out of the four novice users however were able to infer how to run the code without help. Test recordings suggest that they ran LabVIEW code by pressing this arrow before I explained to other users how to do so, meaning they were able to find the run button on their own. In the future, however, a “Helpful Tips” sheet or other form of coaching for novice users should include how to run code on LabVIEW so users do not face this barrier.

IoT Education Toolkit Features

The errors I describe in this section are those participants made in the usability test that prevented them from navigating through the tasks without inaccuracy or misunderstand. These are features that are only available if a user has installed the Toolkit on his computer or laptop. Some of the elements do overlap with features in LabVIEW, but these specific errors have direct involvement with the Toolkit as well.

1. Not finding the ThingWorx login from the Toolkit Express VI window

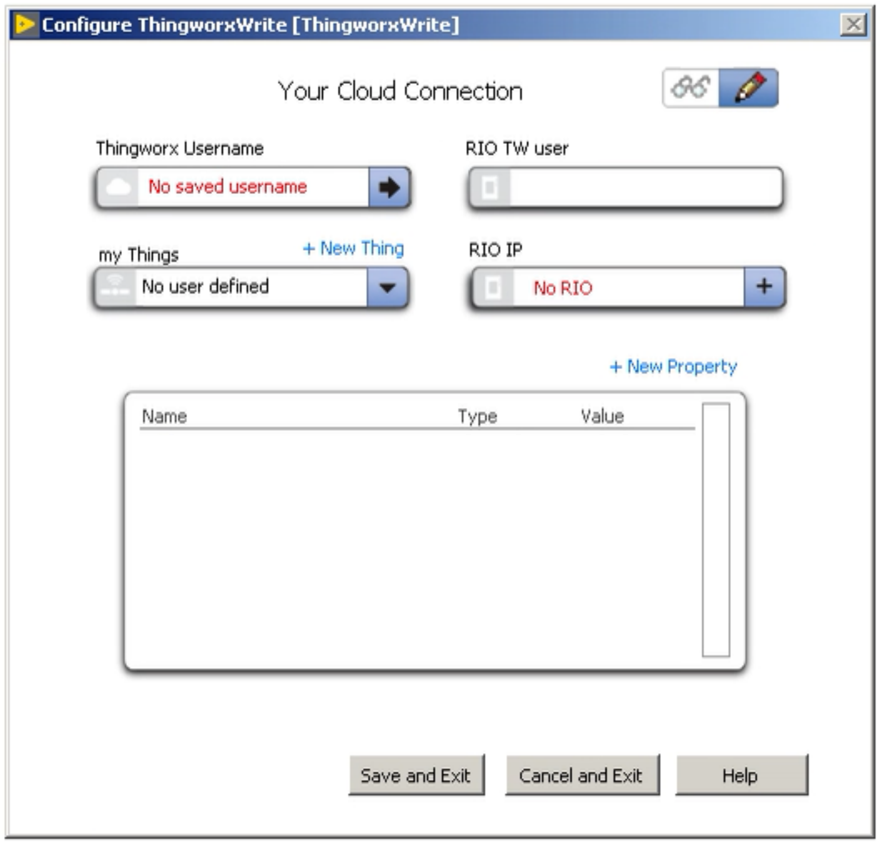


Figure 7.10: IoT Toolkit Express VI Window: Dialogue box for initiating Things and properties for a user's ThingWorx account. A user must select the blue button with a black arrow in the "ThingWorx Username" box to sign into his account and retrieve or create Things and properties

Figure 7.11 above shows the Express VI for the read or write block when a user drops down the VI on the block diagram. To log into a ThingWorx account, a user must click on the blue button (black arrow) next to the red text "No saved username." Three of the participants in the usability study either could not find this login within 30 seconds of opening this window and/or asked for my help to log in.

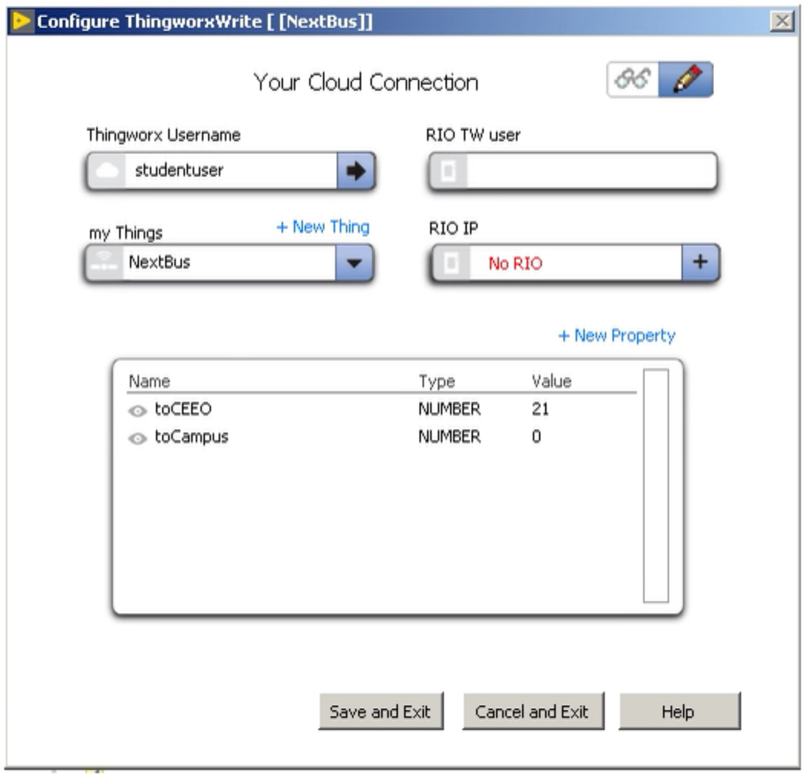
Gap

Participants' errors in this instance revealed a gap between the *Expected Service and the Perceived Service* of the student resulting from a gap between the *Developer's Translation of Perceptions of Consumer Expectations into Service Specifications and the User's Perceived Service.* The developers understood that students would have to log into their ThingWorx accounts intuitively, and so they included a button to access this page (the service specification), but users who failed to navigate to this page on their own or with ease revealed the discord between these two goals.

Improvements

This error suggests a user interface flaw in the IoT Education Toolkit system. For a more effective approach, this window should feature a more intuitive method for the user to access the login page from here. Typically, interfaces will feature next to the login bar a "login" label that will direct users to a login page.

2. Failing to select a property on the Express VI window


Configure ThingworxWrite [[NextBus]]

Your Cloud Connection

Thingworx Username: studentuser

RIO TW user:

my Things: NextBus

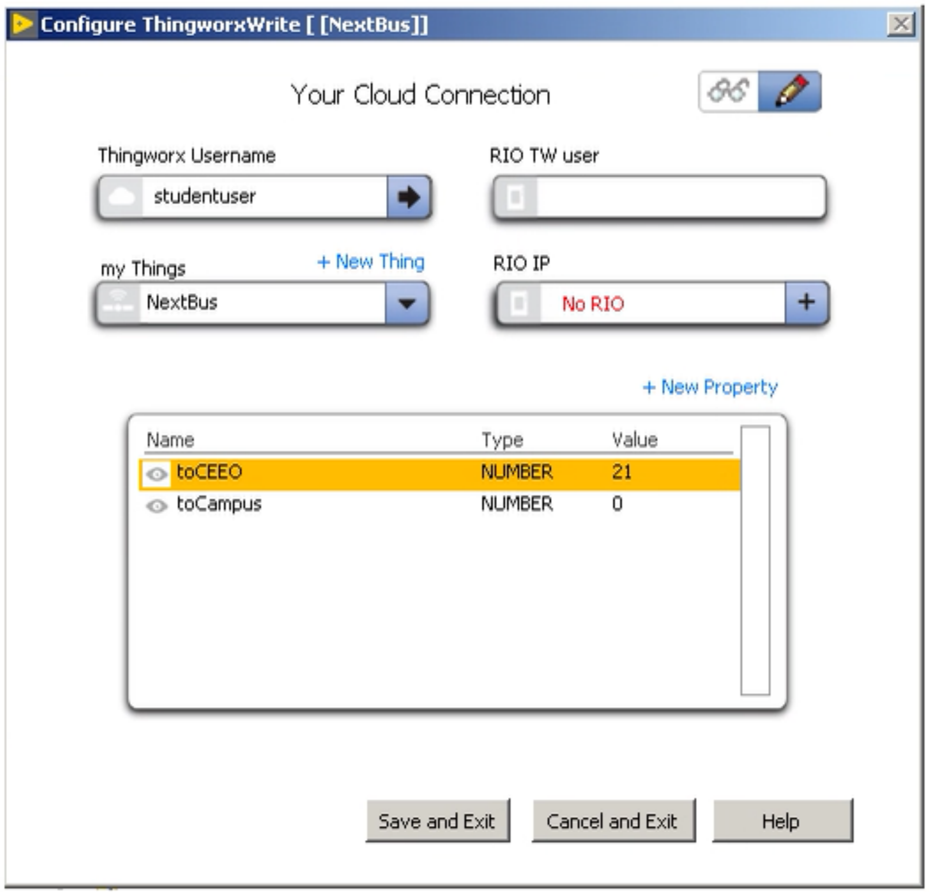
+ New Thing

RIO IP: No RIO

+ New Property

Name	Type	Value
toCEEO	NUMBER	21
toCampus	NUMBER	0

Figure 7.11: No Property Selection: Express VI Window. Dialogue box for Toolkit's Express VI window for the read or write blocks. Thing NextBus has been chosen, but no property value has been selected


Configure ThingworxWrite [[NextBus]]

Your Cloud Connection

Thingworx Username: studentuser

RIO TW user:

my Things: NextBus

RIO IP: No RIO

Name	Type	Value
toCEEO	NUMBER	21
toCampus	NUMBER	0

Figure 7.12: Property Selection: Express VI Window. Dialogue box for Toolkit's read or write Express VI. Yellow bar highlights the property selection

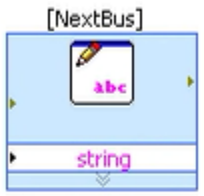
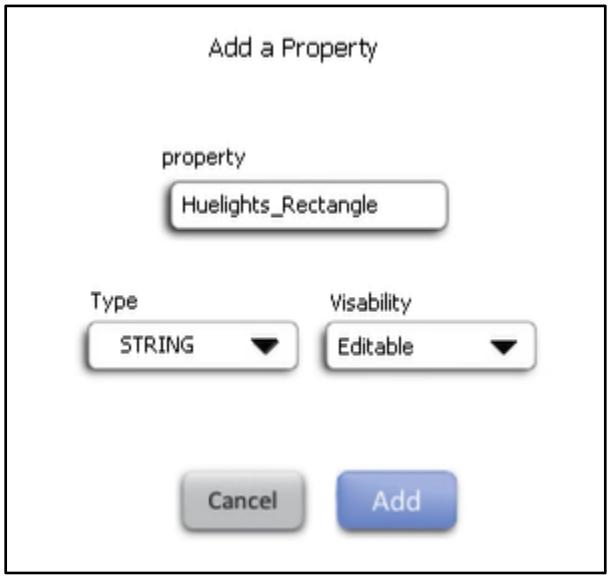


Figure 7.13: No Property Selection: Express VI. Result of user failing to select a property from the read/write Express VI dialogue box. Defaults to selected Thing (NextBus in this example), and arbitrary string value

Figure 7.12 shows the Express VI window that appears when a user selects a Thing from the "my Thing" menu, but does not choose a property. Figure 7.13 shows the yellow bar that gives the user feedback to which property he or she has selected. When a user selected “Save and Exit” from Figure 7.12 (without selecting a property), Figure 7.14 appears on the block diagram, which automatically calls for a string input, even though the two properties in this Thing category asks for numbers. Five of the usability participants made this error at some point while completing their tasks.



property: Huelights_Rectangle

Type: STRING

Visability: Editable

Figure 7.14: Add Property: Allows users to add a property, chose the type of input/output variable (string, boolean, number, etc.), and visibility, or security setting (editable, viewable, or private)

One user also tried to create his own property by selecting “+ New Property” and typing in ”Huelight_Rectangle.” He did not realize at first that he had to select from the given properties.

Gap

Failing to select a property, or creating a new property that already exists are UI faults that developers can avoid with appropriate improvements. The gap here between the *Developer’s Translation of Perceptions of Consumer Expectations into Service*

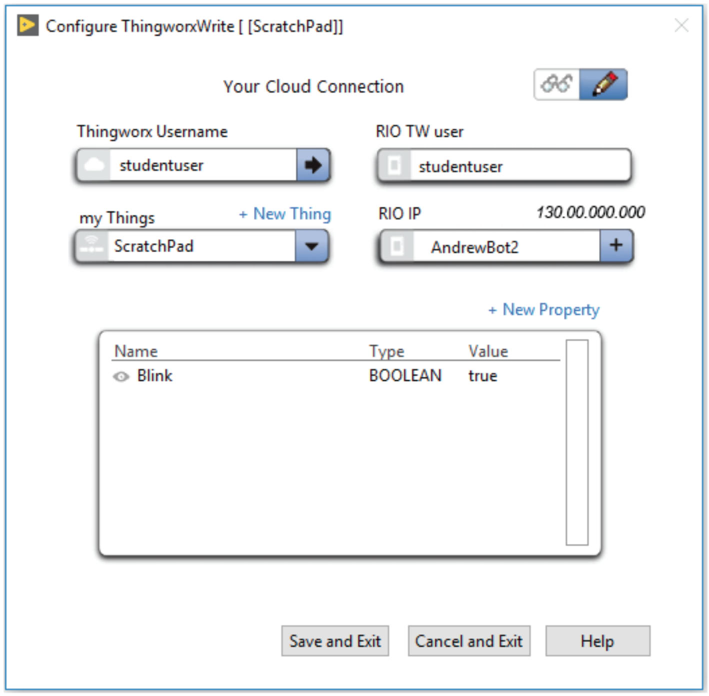
Specifications and the User's Perceived Service, since designers of the platform knew that users would have to select a property or create their own, but failed include the errors that can arise when users interact outside of developer's expectations and fail to select a property, or create one that already exists.

Improvements

The error outlined in Figures 7.13 and 7.14 suggest that feedback here would be helpful for users to realize that they have failed to select a property. If a user presses "Save and Exit" or "Cancel and Exit," a pop-up block that alerts the user that he failed to select a property would help avoid this error.

For the error made in Figure 7.15, a drop down menu with available pre-created properties would have prevented this user from making this mistake.

3. Failing to select a Thing on the Express VI window



Configure ThingworxWrite [[ScratchPad]]

Your Cloud Connection

Thingworx Username: studentuser

RIO TW user: studentuser

my Things: ScratchPad

RIO IP: 130.00.000.000 — AndrewBot2

Name	Type	Value
Blink	BOOLEAN	true

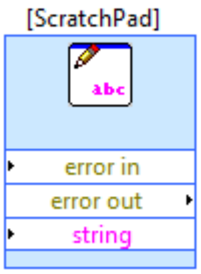


Figure 7.15: No Thing Selection: Express VI Window and Resulting Express VI: When users did not select a Thing from the Toolkit dialogue box (left), but pressed “Save and Exit,” the resulting Express VI defaults to Thing “ScratchPad,” which takes a string value

The above diagrams show an error that 3 participants made during the usability test. They failed to select a Thing from the “my Things” drop down menu before selecting “Save and Exit.” The figure on the right shows the respective Express VI when this occurs. The Toolkit automatically chooses the “Scratchpad” Thing, which the Getting Started Utility uses to show the “blink” gadget to display connectivity feedback. Participants in the study however did not want to write to this Thing in order to complete the tasks.

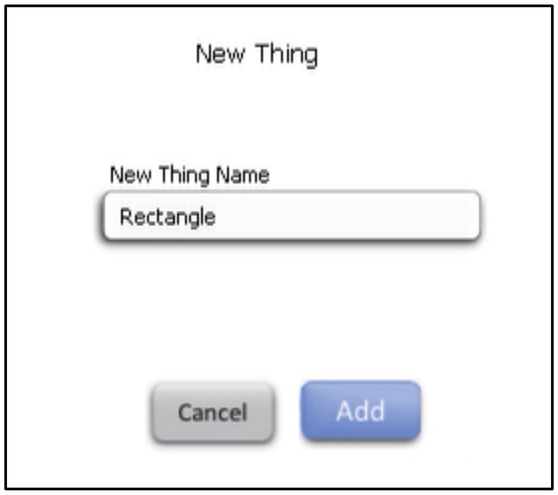


Figure 7.16: Add New Thing: Allows users to create a new Thing

One participant also tried creating his own Thing by selecting the "+ New Thing" text on the Express VI window. The above figure shows how he tried entering "Rectangle" (his assigned name) to the text bar. Having a drop down menu here as well, that illustrated the available pre-created Things with similar names to the one being typed, may have prevented this user from making this mistake.

Gap

The gaps of this error mirror that of the previous error of a user failing to select a property on the Express VI window. The gap lies among the *Developer's Translation of Perceptions of Consumer Expectations into Service Specifications and the User's Perceived Service.*

Improvements.

Similar to the previous error, feedback for the user in the form of a pop-up menu alert for unselected Things and a dropdown menu for a list of already created Things would help prevent these user errors.

4. Using a front panel to create Toolkit controls

My instructions (Appendix C) showed users how to find the read/write Express VI's and from their block diagrams. However, half of the participants (four out of the eight) used the presented navigation trajectory (right click > IoT Education Toolkit > ThingWorx > Read/Write blocks) on their front panels after they successfully opened up the VI from the project. Figure 7.18 below shows the folder that users navigated to when making this error, and Figure 7.19 shows the front panel's resulting diagram.

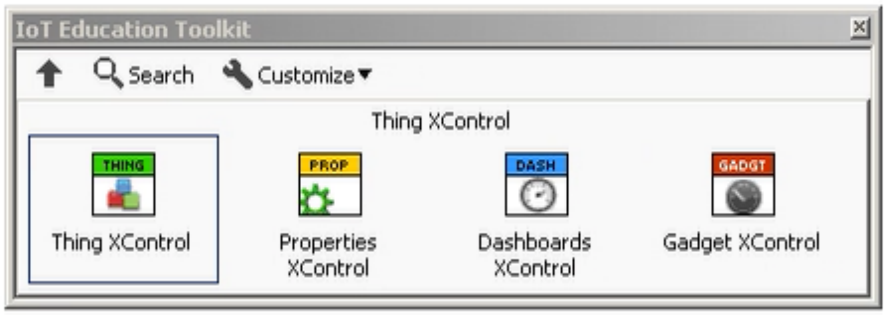
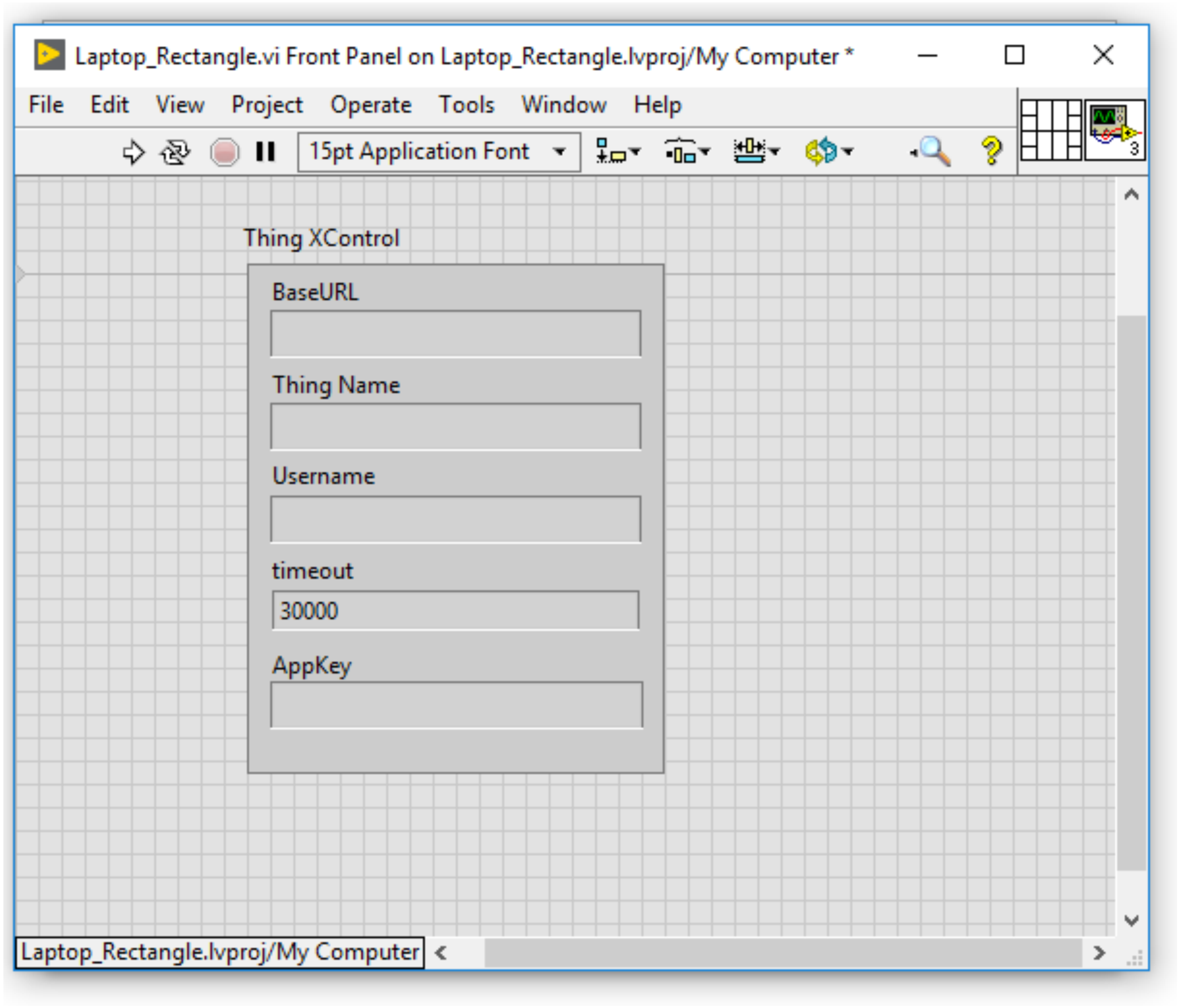


Figure 7.17: IoT Toolkit Menu From Front Panel: Icons for Toolkit library for front panel controls. In user testing, some participants navigated to this set of Sub VI's instead of those presented on the block diagram, as outlined in the usability test directions


Laptop_Rectangle.vi Front Panel on Laptop_Rectangle.lvproj/My Computer *
File
Edit
View
Project
Operate
Tools
Window
Help
15pt Application Font
Laptop_Rectangle.lvproj/My Computer

Thing XControl

BaseURL

Thing Name

Username

timeout

30000

AppKey

Figure 7.18: IoT Toolkit Thing XControl on Front Panel: In testing, some users chose the Thing XControl icon from Figure 7.17 and dropped it down on the front panel. This Figure is the result of that placement

This diagram would not allow the users to create the controls or constants for assigned Things and properties, and therefore prevent them from completing the tasks successfully. When a user first opens a VI from a project, the front panel is the only page that is opened, and he must navigate himself to the block diagram, which serves as the platform to write most of the code. However, during the usability tests, five of the participants had difficulties finding the block diagram.

Gap

This error falls along the gap between the *Developer's Perceptions of User Expectations and the User's Perceived Service*. Personal experience, involving coding experience, feeds into the user's expectations. Developers assumed that even novice users would be able to navigate through the LabVIEW platform.

Improvements

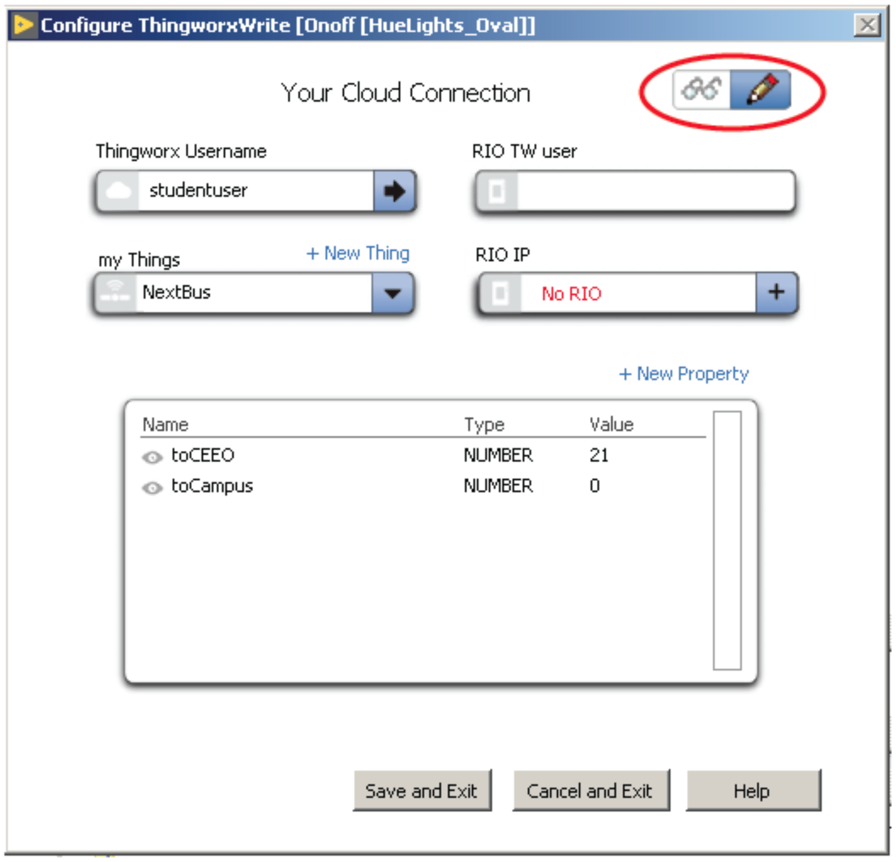
If the block diagram was instead the page that opened to users first, or I had given more strict and clear directions to go to the block diagram first, participants perhaps would have had more success in finding the IoT Toolkit controls as I presented to them. The former consideration is a LabVIEW platform specification, but a Help Guide that pairs with the platform could still highlight these elements.

5. Using the wrong read or write block

Resulting from another UI error, five participants chose an incorrect read or write block to send or receive data from the cloud. The ThingWorx folder contains these two blocks, which I outline in the "Helpful Tips" handout (Appendix C). Although I do not tell participants which block to use for each task, I have keywords in the directions that suggest which one they should choose.

Many users (five of the eight) made at least one error in choosing which block to drop down and use on their block diagram. Some were able to figure out on their own that they had made an error after they realized they could not connect a control or constant to a read block, but some only realized they made an error after I pointed it out to them and gave them a brief overview of how they were interacting with cloud data in the task at hand.

Two participants from two different usability test sessions understood how data was connecting with the data during their tasks (apparent in conversations with me during usability testing and/or through accurate post-task data flow diagrams), but made a mistake of switching the read/write command available on both the read and write blocks (shown below).


Configure ThingworxWrite [Onoff [HueLights_Oval]]

Your Cloud Connection

Thingworx Username: studentuser

RIO TW user:

my Things: NextBus

+ New Thing

RIO IP: No RIO

+ New Property

Name	Type	Value
toCEEO	NUMBER	21
toCampus	NUMBER	0

Figure 7.19: Read/Write Toggle on Express VI Window: Dialogue box of Toolkit Express VI window with toggle switch to change from read or write command

This toggle can change the Express VI from reading values to writing to them (or vise-versa). These two participants seemingly were exploring this button, did not receive proper feedback, and made the mistake of making appropriate selection. Later in the study, I told one of the participants that she made this mistake, and she told me she did not realize that she had selected the wrong figure, even though she knew she was supposed to choose the other one. She also gave feedback on the UI for this feature.

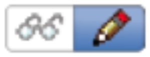


Figure 7.20: Read/Write Express VI Toggle: Icon on upper right side of Toolkit Express VI dialogue box to switch to or from a write/read command

Referring to Figure 7.21 above, this participant states, “I'm guessing this is view and this is write, but I don't know if highlighted or not highlighted is on or off."

The other participant who made this mistake of selecting the wrong icon from Figure 7.21 switched his block to the correct one after I made his mistake apparent to him, but he did not give any UI feedback stemming from his error.

Other errors revolving around this read/write block discrepancy involved cognitive misunderstandings of what each block does. For example, I had the conversation below with a user after I noticed she was using the write block instead of the read. She asked for help after she tried and failed to have a NextBus property value change her Hue Lights Strip. I had her explain to me how she thought her code should be running and what each VI and connection meant. She had created a while loop around a write Express VI that would write to a Next Bus property.

UP2: “This [NextBus vi] gives values of 9 and 8…”

Interviewer: “So, just to back up… you say it’s giving values, but you have it under the write [vi].”

UP2: “But don’t I still have to write it to change the color?”

I explain to her that since she’s reading numbers from the cloud, she’s going to want the read VI to receive the numbers, and also the write VI to change the color.

Furthermore, I noticed that two of the participants, when trying to switch from a read to a write block or vise-versa, would delete the blocks and choose the correct version in the ThingWorx folder instead of using this toggle button and switching to the desired icon.

Gaps

There are two gaps that arise from the read/write discrepancies. One is a UI error, which stems from the *Developer’s Translation of User Perceptions into Service Specifications, and the User’s Perceived Service*. The designers tried to create the toggle in Figure 7.21 in the Express VI window to allow users to switch between read/write commands easily. However, it caused users in the usability test to make errors and therefore failed to meet developer’s goals.

The second error in which the user did not understand the difference between read and write, emerges from a gap between *the User’s Expected Service and the Developer’s*

Perceptions of Consumer Expectations. Although I provided some guidance to which block the users should be placing on their diagrams for each task, this notion of reading and writing from the cloud still caused confusion and needs further explanation before giving these tasks to novice users.

Improvements

The lack of understanding the difference between the read and write block and the utility of the toggle switch could be solved through appropriate coaching and Toolkit descriptions or help resources.

6. Running the code from the wrong source (PC vs. myRIO)

Three participants made the error of running their code from their laptop when they were supposed to run it from the myRIO, or vise-versa, per the task's description on the assignment handout. I state in the pre-test video that I had created two projects with their own VI's, one on the laptop, and one on the myRIO. For the first two tasks, I highlighted the difference between the two to make sure that users understood where they were running their code (Appendix B). However, three participants still made this source error. These errors varied from failing to switch from the laptop to the myRIO, or only running on the myRIO and not the laptop. I will go into further detail on myRIO in the Adding Complexity section.

Of these three participants who made these errors, none of them created a correct post-task diagram for the second task, which sends data to the cloud from the myRIO. This suggests that their errors were not just stemming from a failure of following the directions on the handout, but also a lack of understanding on how the myRIO works and pairs with the platform.

Gaps

The gap in this error branches from *the User's Expected Service and the Developer's Perceptions of Consumer Expectations.* The developers expected users to be able to perform this simple command, but user's prior experience (or lack thereof) prevented them from not only realizing where they were running their VI's did not match that of the tasks, but also hindered them from understanding what "running from the myRIO or laptop" actually meant.

Improvements

Students need help understanding the purpose of the microcontroller, its benefits, and the meaning behind running a code through it compared to a computer. Having information sessions through short videos (that are more informative and thorough than my own) could help reduce this confusion. Also, having feedback on LabVIEW that informs the user where they are running code could also help make this process and data trajectory obvious to the user.

7. Not being able to navigate to the Express VI window after closing it

After a user exists out of the Express VI window by either pressing "Save and Exit" or "Cancel and Exit," he or she can open up this window again by double clicking on the Express VI from the block diagram. Two participants, when trying to open this window again, failed to realize that they could navigate to the window by this method, so they deleted their Express VI's and opened new ones from the ThingWorx folder. Both of these participants did not have LabVIEW experience, suggesting that this method may come natural to users who have prior experience with the platform, but may not be intuitive to those who do not.

Gaps

Although this may not be a tragic error that prevents a user from performing tasks, it is still an unnecessarily long path that the user follows, which can be cut shorter for a more efficient user experience. I classify this gap as being between the *Developer's Perceptions of User Expectations the User's Perceived Service.* Although the Express VI allows for users to double tap its block to open up the Express VI, developers do not make this feature obvious and fail to make this feature obvious to novice users.

Improvements

Adding this "double tap" method to open up the Express VI window should be included in a LabVIEW/ IoT Toolkit Education guide, so users can access their data sources more efficiently.

8. Wanting to make a cluster as an input/out for a write/read block or have write blocks accept more than one input

In plain sub VI's and Express VI's in LabVIEW, users may be able to add several different input or outputs around the block, as shown below.

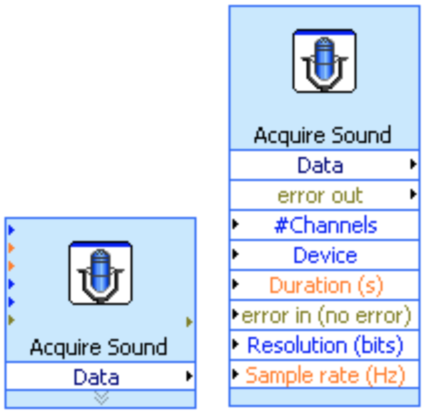


Figure 7.21: LabVIEW Example Express VI: Default setting (left) and multiple inputs and outputs on expanded form (right)

For the Express VI in the IoT Toolkit, there only features one node for an input or output, regardless if the user only selects a Thing without selecting a property, or tries to select more than one property (see figure below).

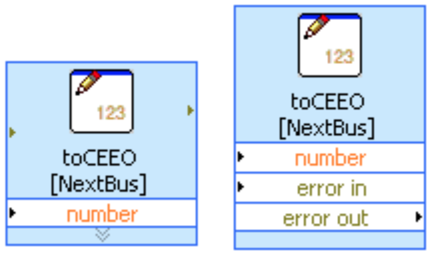


Figure 7.22: IoT Toolkit Express VI: Default setting (left) and expanded form (right)

Post-task interviews, screen recordings, and conversations while users were navigating in the platform revealed that three participants either tried selecting more than one property on the Express VI window, or expanding their Express VI's to try to show all of the available properties for the selected Thing, as shown in Figure 7.23 above. Two of these participants had prior experience in LabVIEW, suggesting that they were aware that this feature is available in LabVIEW for other VI's, and assumed it would be included here as well.

Adding this feature to the Toolkit's Express VI's would allow users to code their diagrams more efficiently by reducing the number of VI's needed on the block diagram. For example, users may want to edit the brightness and the color of their Hue Lights Strips, but to do so, they would have to write to two different Express VI blocks.

Gaps

The gap here falls between *Development's Perceptions of User Expectations and User's Actual Expected Service.* Developers failed to recognize this feature as a user expectation and therefore did not form service specifications, even though some users expected this expansion feature to be an option to add more inputs and outputs.

Improvements

Creating this feature of expanding the Express VI to add multiple property output/input values from one Thing would solve this empty user expectation.

7.2 ThingWorx Dashboard

The dashboard that pairs with the IoT Education Toolkit offers novice users a way to "look into" the cloud to view the data that Things are pushing to or pulling from. Some students reporting using the dashboards developers had intended, while others did not. I outline the gaps that evolve from this feature in this section.

7.2.a. Developers

According to the PTC developer, the users who did view the online dashboard did so according to his expectations for the first level of complexity. They used the webpage as an affirmation of running a successful IoT program, or used it as a tool to debug their system.

The developer was also hoping that students would be able to create more complex systems in the dashboard. He hoped that "...students would actually create some basic applications that would be meaningful, so that [data] would interact at a complex enough level so that they could control something." However, this additional second layer of complexity was never implemented in the Toolkit, so students were not given a change to create these more intricate data analytics in the cloud.

7.2.b. Users

Collectively, about half all participants in the usability test and student interviews stated or purposely navigated to the ThingWorx dashboard at least once. Three out of the five students from the robotics class stated they at least once opened the dashboard, and all of these users said they used it, or viewed it, as a page that showed a successful piece of data flowing to or from the ThingWorx cloud.

Two students from the class expressed their opinions on the limitations of the dashboard however. One states that he didn't "entirely understand [its] purpose when Composer exists," suggesting that Composer not only displays data like the dashboard does, but it also provides a platform to create more complex controls. The other student, who ran into the dashboard's boundaries, stated that he was hoping to view the JSON string that he was sending to the cloud, but he could not.

Four out of the five usability test participants also used the dashboard, and understood its objective, according to think aloud transcripts, conversations during testing, and screen recorded data. These participants also used the dashboard as a means of viewing successfully code. Although the screen recordings did not reveal the participants' screen clicks, a think aloud quote from one user suggests that she was expecting this page to be interactive. When navigating to the page for the first time, she says, "Ok, so this must be feedback, so you don't actually change [values] here." It is unclear if other participants held this expectation as well.

Users probably all utilized the dashboard in the same manner since the webpage offers minimal opportunities for user interaction, it merely displays data.

7.2.c. Gaps

The three major gaps that separate the current and ideal state of the dashboard are as follows:

1. Developers failed to *Translate Perceptions of Service Specifications into the Actual Service.* The PTC developer who helped create the dashboard illustrated his original goals that he hoped the students would utilize in the dashboard, but offered no coaching or instruction on how to do so. His failure to introduce these additional data analytics could be attributed to time constraints.

2. The *Actual Service does not Align with the Consumer's Expected Service*. The two robotics students who I interviewed, and one of the usability participants realized the limits of the dashboard, and were expecting a platform that would allow manipulation of the cloud data. While ThingWorx Composer offers this feature, that system requires too much coaching for a novice user to the platform.

3. The *Consumer's Perceived Service and the Actual Service do not match*. This gap centers around the six out of thirteen total participants who reported never using the dashboard. These users perceived a Toolkit without this feature, as they either ignored that it was there, or had never seen it before in the case of two of the robotics students. This could be attributed to a partner's inclining to accessing the dashboard, never needing to debug, or forgetting that the dashboard existed and what featured.

7.2.d. Summary

Mid-way through the Fall 2017 semester, the professor tried to introduce the ThingWorx terminal, which offers a more involved method to view and edit data, Things, and properties in the cloud, but reportedly, students did not use this feature either. This could be attributed to the fact that they were used to the current design. Further user testing with participants with LabVIEW experience would reveal its usefulness and if it could possible replace the dashboard as a more appropriate method to view and edit cloud data that does not involve all of the training that Composer requires.

7.3 Debugging

Students need a way to fix their IoT systems when they fail to run properly. Appropriate feedback and tools would make this process easier. LabVIEW provides a set of tools including “probing,” as well as its help menu. I outline the probing method in the Results section.

The IoT Toolkit contains the dashboard, Composer, and terminal, where students can view the cloud they push or pull to or from the cloud. I have outlined all of these features in previous sections, but to reiterate, the dashboard allows students to view data from a web page, Composer lets users view and manipulate online data via web page, and the terminal allows users to view and edit cloud data from the LabVIEW menu.

7.3.a. Developer Interviews

The PTC developer said that that “...right now the only way to really troubleshoot that is through ThingWorx Composer which has some challenges but if you're trained on it it's definitely possible,” but since most students were not trained on this platform, it was an inappropriate method to debug the cloud data. He went on to suggest, “maybe bringing some of those strategies that exist in ThingWorx Composer down to the Toolkit level so that they're more accessible.” He also mentioned that while the dashboard gives students a way to see their data successfully reach the cloud, there is no clean display of feedback when something is not sent, or where “Things go wrong.”

7.3.b. Student Interviews and Usability Tests

Students revealed in interviews and usability tests that to debug their cloud data, they either create a read block that is reading data they were hopefully pushing to the cloud, look in ThingWorx Composer, use the LabVIEW probing method, and view cloud data through the online dashboard. One participant from the student interviews revealed that she had prior experience with Composer, and used it to debug. She also taught one of her partners how to use some of the features in Composer, who represents the second participant who reported also using Composer during his projects.

The robotics class students also had a difficult time initializing their myRIO's, especially when they were changing partners and using

7.3.c. Gaps

The gap that became apparent in the system when discovering how students debug their projects stems from ThingWorx *Developers Failing to Take Perceptions of Consumer Expectations and Making a Translation of Those Perceptions into Service Specifications*. In other words, they did not create an intricate enough dashboard or another element to allow students to effectively debug cloud data. I discuss in the next section, "Adding Complexity," of a feature that serves as a combination of the online dashboard and Composer, which would allow students to few and manipulate data. This would also help students debug their systems, and if developers do consider putting a system like this in place, they should add feedback to show students where the system is failing if data is not reaching the could.

Another gap lies between the *Consumer's Expected Service and Actual Service*. They expect clear feedback from a system to understand which software is installed on their devices and which Wi-Fi network is it connected to, if any. The myRIO has an onboard LED that shows *if* it is connected to Wi-Fi, but the actual network can only be seen through an online web page. Having a cell phone app or page with a more straightforward interface that students will be more likely to use, would allow students to set up their hardware more easily and diminish the time it takes students to prepare myRIO for the first time, or again after switching partners.

7.4 Adding Complexity

Ideally, novice users should be able to create simple IoT projects quickly, like the tasks users performed in the usability tests, but once they become more comfortable with LabVIEW and the platform, students like the ones in the robotics class, have access to expanding on the more advanced technologies of the myRIO and ThingWorx systems.

7.4.a Developer Interviews

PTC

This IoT Toolkit developer stated that he hoped students would have a reason to explore the ThingWorx Composer platform in order to create a more intelligent IoT project by adding a layer data analytics. However, he realizes that this is a complicated process, as there are hundreds of sources of data for students to pull from, but no common process or platform to do so. He further explains that pulling the same type of whether data will look different if a student tried drawing it from weather.com or Wunderground (two different online weather data sources).

NI

The NI developer understood the abundance of allowances that the myRIO provides, and was aware of the price that is outside of a typical undergraduate student's budget, but he discussed that the myRIO is a viable option to pair with this Toolkit since it allows students to create simple and more advanced robotics projects. Although its capabilities exceeded a typical undergraduate robotics classroom, it does serve as a suitable platform for students to learn more about the advanced technologies the myRIO offers like the FPGA system.

7.4.b. Student Interviews

Students from the robotics class did acquire data from third party sources in their weekly projects through Toolkit features, but they did not explore ThingWorx Composer to investigate how to add advanced data analytics.

They had few issues with the myRIO, but were curious about how other microcontrollers would fit with the platform that were not as "bulky" or robust. The problems they did encounter with this piece of hardware were in loading the latest image, or software version, and connecting to the Wi-Fi network. Connection issues, one student explained, could be due to the fact that the University Wi-Fi system can be difficult to connect to. However, this student also said he wished there was an easy platform to view which Wi-Fi the myRIO was trying to connect to or already connected to, besides the online web page it offers. Three of the five participants did mention in the interview that they enjoyed learning about or found useful, the FPGA system of the myRIO as it allowed them to make their projects run faster.

7.4.c. Usability Tests

I explained some of the myRIO's logistics and features in the post-task video I shared with the usability test participants, but they only used the myRIO button control as an integration with the tasks (besides also using it as a computer to run code in Tasks 2-4). Even if these students did not fully understand all of the myRIO's capabilities, a few of them seemed to view the microcontroller as a highly capable piece of equipment. One participant, who had prior experience in LabVIEW, said in the post-test survey that she was curious how the myRIO would "enhance IoT systems," and how it sets itself apart from a myRIO. This curiosity suggests that students interested in IoT are inherently interested in its more advanced hardware technologies.

Summarizing some of the post-task diagram figures, six of the eight participants created a correct data flow diagram, but when they continued to Task 2's assignment, two of these participants failed to create a proper display of how the data was traveling through the system. This suggests that adding the myRIO or other microcontroller to an IoT system may need additional explanation in order for students to understand how it fits into the system. Having this basic awareness will allow users to scaffold their IoT systems better, giving them opportunities to explore the myRIO's more advanced capabilities.

7.4.d. Gaps

Based on interview data, I gather that there is one major gap in allowing students to add complexity to their projects in this platform. It falls between the *Developers Taking Perceptions of Consumer Expectations and Making a Translation of These Perceptions into Service Specifications*. The developer mentioned the power in creating data analytics and machine learning algorithms in ThingWorx Composer, but understood that there was no easy way for students novice to the Toolkit and ThingWorx to do this in a classroom setting without time-consuming training.

For future iterations of the Toolkit, developers should consider creating an element that gives students an opportunity to create data analytics from features that ThingWorx provides on its Composer platform. Furthermore, the dashboard, as I outline in previous sections, limits the students capabilities from manipulating data like Composer allows. The ThingWorx terminal provides a way for students to view and edit data, but students failed to use this platform as it was introduced mid-way through the semester in the robotics classroom and I did not present the feature in the usability tests. Developing this terminal more to match some of the more advanced capabilities of Composer would give students the benefit of having more advanced capabilities while not needed to be "a fully trained software engineer," as the PTC developer explains.

The myRIO served as a fairly acceptable microcontroller for the Toolkit, especially since it pairs well with LabVIEW. It has more advanced specifications than students may need, and does not serve as a piece of hardware that students would want to buy on their own, but as a learning tool, it is suitable for undergraduate classrooms. If developers were to consider creating another microcontroller to pair with the Toolkit, they should include the following elements, some of which the myRIO already contains:

- Easy installations
- Wi-Fi feedback
- FPGA system
- Accelerometer
- Small size

7.4.e. Summary

In general, the platform gives students the abilities to make additional scaffoldings to their simple IoT systems and create more advanced designs. The only feature the Toolkit lacks is the ability to make their projects “smart” by adding analytics that would take changing information from third party sources and create machine learning algorithms that would make decisions for their robots.

7.5 Sharing data

7.5.a Developer and Student Interviews

The interviews with students and developers revealed that the expectations from both parties were that students in the robotics class would be collaborating with each other frequently, and since they were not to be sharing sensitive data, privacy settings could be kept minimal. Throughout the class, this expectation held true, as students were constantly working with partners on weekly projects.

However, this sharing of data created a problem for students when reportedly a student deleted properties on the cloud by accident. This problem was resolved, but it still raises a concern for additional privacy settings, or creating a UI with more feedback to ensure that a user does not delete or overwrite a property by mistake.

7.5.b. Gaps

A gap in this sharing data component lies between the *Perceived Service from the Consumer's Perspective, and the Actual Service.* The developers created the platform without the additional considerations of students accidentally manipulating each other's data. User interviews suggest that this was a mistake and not a malicious act by another student, but to bridge this gap, one feature developers could consider is having additional securities settings for each laptop or myRIO IP address. This adds a layer of complexity that could prevent users from efficiently collaborating with each other however, so another option could be to give users more feedback on the user interface to prevent these mistakes.

7.6 Future Conditions on Approaches

Although the interviews and usability tests revealed important gaps in the platform, I would approach the assessments differently if I were to conduct these analyses again. First of all, I would either study students in the robotics class during their semester as they use the toolkit, and closely examine their relationship with the platform. For example, if they chose not to use the toolkit for a specific weekly project, I would question their motives. This approach would allow me to uncover more of the "Adding Complexity" limitations, as well as determine first hand the in-class issues that students had reported with the Getting Started Utility and sharing data. I conducted interviews with the robotics students 2-3 months after the students had completed the robotics course, so performing studies and/or interviewing students while they are participating in a class with the toolkit would make it easier for the students to recall their projects and thoughts.

I would also have designed the usability tests differently. None of the participants attempted Task #4, and many did not complete Task #3. I would have made each test an hour and a half long to make sure all participants could complete Task #3. I would be curious to see if the participants who did not have prior LabVIEW experience could complete Task #3 on their own after having gone through the first two Tasks in LabVIEW. Furthermore, I would include a pre-task assignment that would determine how much prior knowledge the participants had with IoT devices. For example, I would show them an advertisement of an everyday IoT device such as a Nest Thermostat, and ask them to illustrate, using given graphics, how it worked. This assignment could establish how well they could grasp how the Hue Lights Strip works and potentially be a predictor of the post-task diagram assignments.

Chapter 8 - Summary & Conclusion

Many of the issues that the robotics students reference in interviews revealed obvious points of concern with the Toolkit and natural hindrances that surface when using developing technologies, like time delays and network errors. Some of the needs of the Toolkit however can be avoided through future iterations of UI design within the Toolkit itself.

8.1 LabVIEW Experience

The most alarming finding that surfaced from the usability tests was that even when users had coding experience in other languages, it was difficult for them to navigate in LabVIEW and therefore complete the tasks. Not only did have trouble physically working through the tasks, but cognitively, they were less likely to understand the flow of data through the system. This lack of awareness could be do to cognitive overload. After all, several users made mistakes that could have been prevented if they listened and watched carefully to the pre-test video and read over the Task Assignment and Helpful Tips handouts.

Prior experience in LabVIEW was the greatest predictor in determining if a usability participant would complete successful attempted post-task data flow diagrams. If developers want to create a successful second iteration of this Toolkit for Universities who do not have the resources that Tufts University did (hand-on help from experts in the platform), they must especially consider this realization and develop a method to coaching students through at least basic LabVIEW features.

At a minimum, a help menu, document, video, or a collection or series of the sort should feature functions like how to open and create a project from the myRIO and computer, and what each of these machines signify, and how to develop a basic piece of code from a block diagram that contains while loops, case structures, etc. Many of the errors that usability test participants made would diminish naturally with more experience in LabVIEW, but there is still this obvious barrier to entry that prevented students from creating and understanding basic LabVIEW services.

Furthermore, this LabVIEW experience vs. lack of experience finding poses another question of whether LabVIEW is an appropriate platform for novice users to create IoT solutions for the first time. All of the participants in my usability study had experience with and "were comfortable coding" in other text-based environments. It raises the question of whether or not these participants would have completed successful post-task assignment diagrams, acknowledging their understanding of the data flow, if the IoT Education Toolkit was based on one of these text-based platforms.

8.2 IoT Education Toolkit UI Improvements

LabVIEW is a strongly established system, and developers of the Toolkit could not have direct control over changing its features for a rapid update. However, designer could help reduce usability errors that the Toolkit presented to participants in the usability test and robotics class.

First of all, making the Getting Started Utility easier to use and understand is an essential feature that the Toolkit is currently lacking. There are different paths developers could go to develop this feature. They could create a getting started experience that involves teaching students about IoT, the platform, and the coding environment, or they could create it more as one of its original ideal forms as a getting started wizard whose sole purpose is to initialize accounts, credentials, software, and networks. Regardless of the direction, developers must keep in mind that novice users to the platform and LabVIEW need time adjusting to the system to be comfortable with the environment, and should provide introductory information (via documents or video's) so users get a better understanding of how the IoT structure works.

8.3 Dashboard/ThingWorx Composer

The robotics students expressed an interest in an intermediary between ThingWorx Composer and the ThingWorx dashboard. Dashboards are useful for novice users to see their values reaching the cloud, but beyond this level of expertise, the dashboard fails to allow users to expand on their knowledge of an IoT platform the way Composer does.

Adding scaffoldings to Composer to make it as intuitive as the dashboard, but offering more features, would give students an opportunity to learn about more advanced concepts like data analytics. The ThingWorx terminal, which I did not formally assess, offers some of these transitional features by allowing users to create their own gadgets on their dashboards, access and manage their Things and properties, and open up examples of IoT solutions, all within LabVIEW. Reportedly however, the robotics students did not take advantage of this element, presumably because they already had methods of maneuvering through the platform to achieve their project objectives by the time their professor introduced it to them.

8.4 Debugging

Interviews and usability tests revealed that users with experience in LabVIEW will use the “probing” method to debug in LabVIEW before trying to access cloud data via dashboard or Composer. Creating this more advanced system I outline in the previous section (8.3) could also create opportunities for users to be able to debug their systems more easily.

8.5 Adding Complexity

This “advanced dashboard” system could also allow students to create more complex solutions without having to be trained on ThingWorx Composer. The fact that the robotics did not use the ThingWorx terminal, which could guide students in developing some of these advancements, suggests that the design should be evaluated, or facilitators of the Toolkit must present this new component with detailed descriptions and/or demonstrations so users can gauge its capabilities.

8.4 Sharing

Developers justifiably avoided complex security settings to avoid impediments in allowing students to share their data across users. This open structure caused issues in allowing other students to delete Things and properties that were not their own, however. Improving security settings enough to share, but not accidentally delete or override Things or properties could take the form of drop down menus of previously created elements to avoid this issue in the future while keeping privacy settings open.

Appendix

Appendix A - Usability Test Transcripts

Script after participants sign the consent form and complete the pre-task assignment

Thank you for participating! You've been given a myRIO, Hue Lights Strip, and laptop computer to complete 3 tasks, which I will present to you after you watch this video.

Video Transcript

The Internet of Things or IoT is a recent buzz topic that is especially relevant to engineering undergraduate students taking robotics and mechanical engineering courses. For industry it allows companies to develop themselves to make their systems more efficient and advance their products through user feedback and data analytics, both of which can be achieved through digital monitoring and IoT. IoT has also recently entered domestic life with smart home products like Amazon's Alexa and Philips' Hue Lights. National Instruments or NI, a producer of automated test equipment and virtual instrumentation software, and PTC, a computer IoT and software service company teamed up with Tufts University to create an IoT education Toolkit software to use in an undergraduate engineering robotics classroom this Fall, 2017 after its beta release in June 2017. This Toolkit runs on NI's coding platform, LabVIEW, and uses PTC's ThingWorx Cloud Services and NI's myRIO, an Internet enabled device, so students can build real-world IOT applications in the classroom.

The NI myRIO is a microcontroller device equipped with complete FPGA coding framework which helps run complicated coding structures more easily, as well as an accelerometer, which detects the myRIO's motion, orientation as well as input/output controls. It runs on LabVIEW and also has four onboard LEDs.

LabVIEW features a graphical programming approach to initializing and analyzing data measurements and controls. LabVIEW uses virtual instruments, VI's, to develop sub programs for a cleaner and simpler graphical program structure. A VI can have inputs, which are controls and constants, and outputs or indicators, to respond to the piece of code it is referring to or running. Here is an example of a sub VI I created represented by a small square labeled 'temp' which converts degrees Celsius to degrees Fahrenheit.

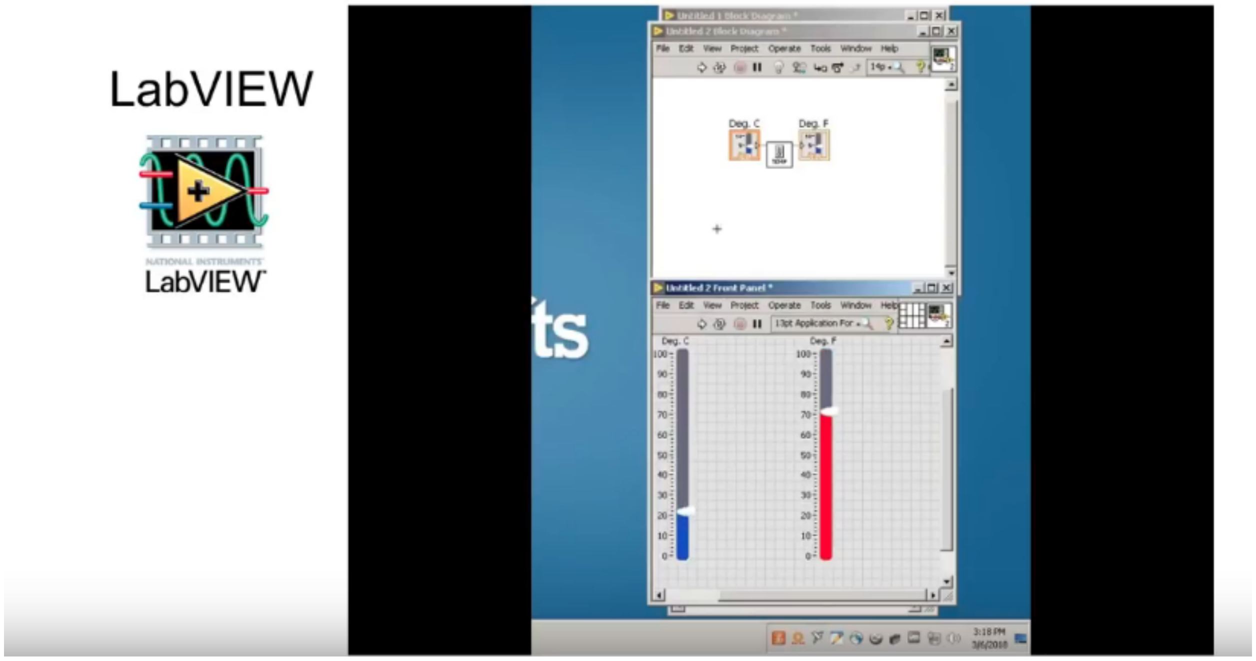


Figure A.1: Example LabVIEW Code

ThingWorx is a cloud-based application development platform used in this Toolkit to store IoT data. In ThingWorx, a Thing is an entity or physical object that has a unique identifier. A myRIO could be an example of a Thing. A Thing can also have Properties, which represent specific characteristics or data from a Thing. For example, the RIO's button status, or an X Y or Z orientation position are examples of Properties of the myRIO Thing.

Going back to the Toolkit, to connect your data you can take your code and attach it to an Express VI if you right-click on the front panel, and navigate to IoT Education Toolkit > ThingWorx > and then you can choose the read or write block. Then you can drop the block on your diagram. You have to provide the username and password which I've provided for you on your printout. If you're running the program from the myRIO you can connect up to the device here as well. Then you can start connecting data.

If you'd like to start a new data source, you can create a Thing here or choose one already created. Here, I use temperature to represent a Thing. You can then create properties. Select your property, give it a name, type, and visibility. Select your property and press save, and you can connect the VI's nodes to your data.

You can access your online data from the ThingWorx dashboard shown here. You can also find this link on your handout, and there should be a webpage on your laptop that is open to this page. You may need to refresh the page to see Things and properties appear.

I got you started and created two LabVIEW projects with new VI's. One running on your laptop computer, and the other running on your assigned myRIO.

Thanks for watching! You can now get started on your first task.

Script after participants finish watching video:

You can now read over the tasks and begin working on them. Please start with the first task, then move to the second, third, and then fourth. I may stop you early so you have time for the post-task assignment and survey.

Throughout the tasks, you may refer back to the YouTube video you just watched, or to any Internet source for additional help. I have also provided a sheet that offers additional LabVIEW tips.

Lastly, these tasks will ask you to change the Hue Lights values or read time information for the next buses that are arriving from or to the Tufts Campus. On this computer [point to computer], *I am reading the Hue Lights properties that you will write, and then controlling the Hue Lights values. I am also reading the bus time values from a 3rd party online source.*

If you have no questions, you may begin the first task. You are encouraged to think aloud as you work through the tasks.

Appendix B - Task Assignment Handout

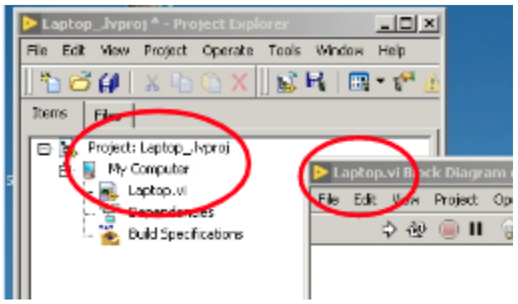
You have a computer, myRIO and light strip assigned to you. The name of your computer, myRIO, and light strip are all the same.

Assigned name: [enter assigned shape name]

ThingWorx:
Username: studentuser
Password: Fall2017

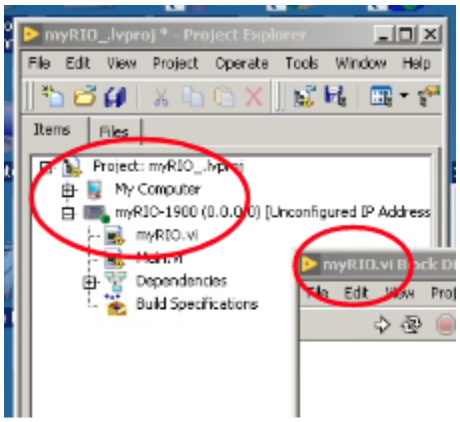
myRIO
Username: admin
Password: (no password)
IP address: [enter IP address]

Link to online dashboard: http://bit.ly/2gwQ3sX

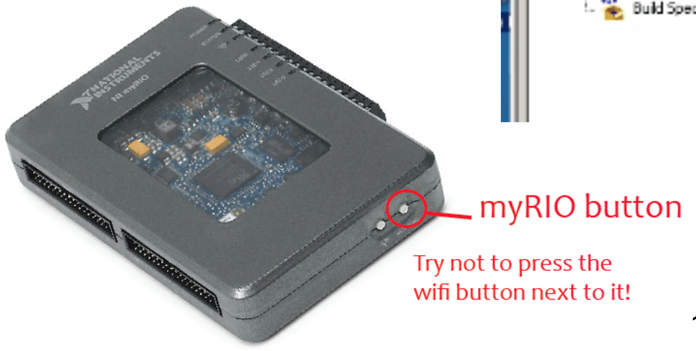


Task 1
Write to the Thing called "Hue lights_yourassignedname" from your computer.
Change the property color from the front panel in LabVIEW. Make sure you're running from your **laptop** (see figure to the right)

Available colors:
red, orange, yellow, green, blue, purple, pink, white
not case sensitive



Task 2
Make the button on the myRIO change the state (on/off) of your Hue Lights Strip. Make sure you create a while loop around your code. Make sure you're running from your **myRIO** (see figure to the right)
See Helpful Tips sheet for where to find a myRIO button block in LabVIEW and how to make a while loop if you need help



Task 3
Pull the property "Next Bus" and control the color, brightness, **or** state (on/off) from the myRIO based on how close/far away the next bus is.
This property gives you the time of the next MBTA 94 bus leaving from Boston/College Ave going toward the CEEO, or vise versa.
Brightness range is from 0-254. Turning the brightness to 0 will turn the hue lights off.

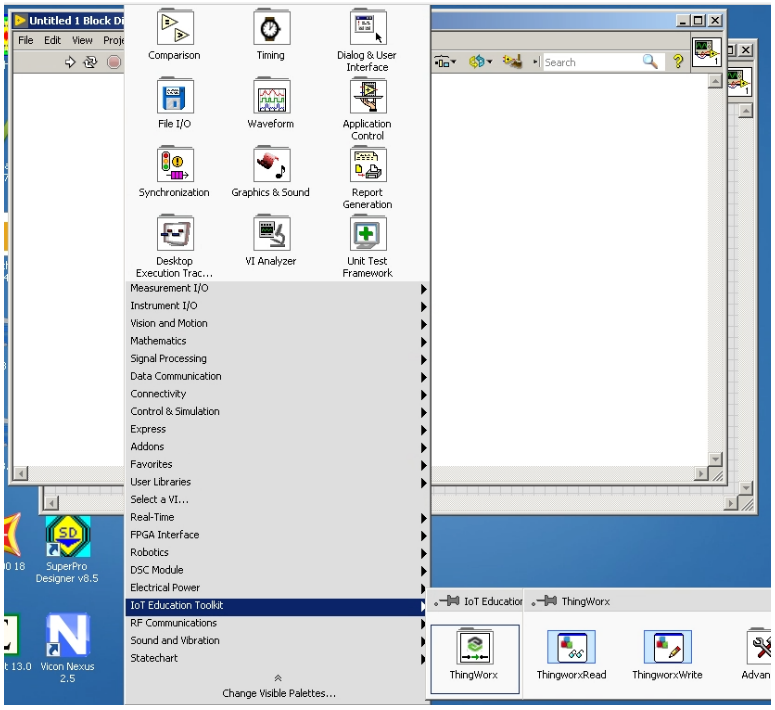
Task 4
Simultaneously, have the computer control the color of your light strip and the myRIO control the brightness or state (on/off).

Appendix C - Helpful Tips

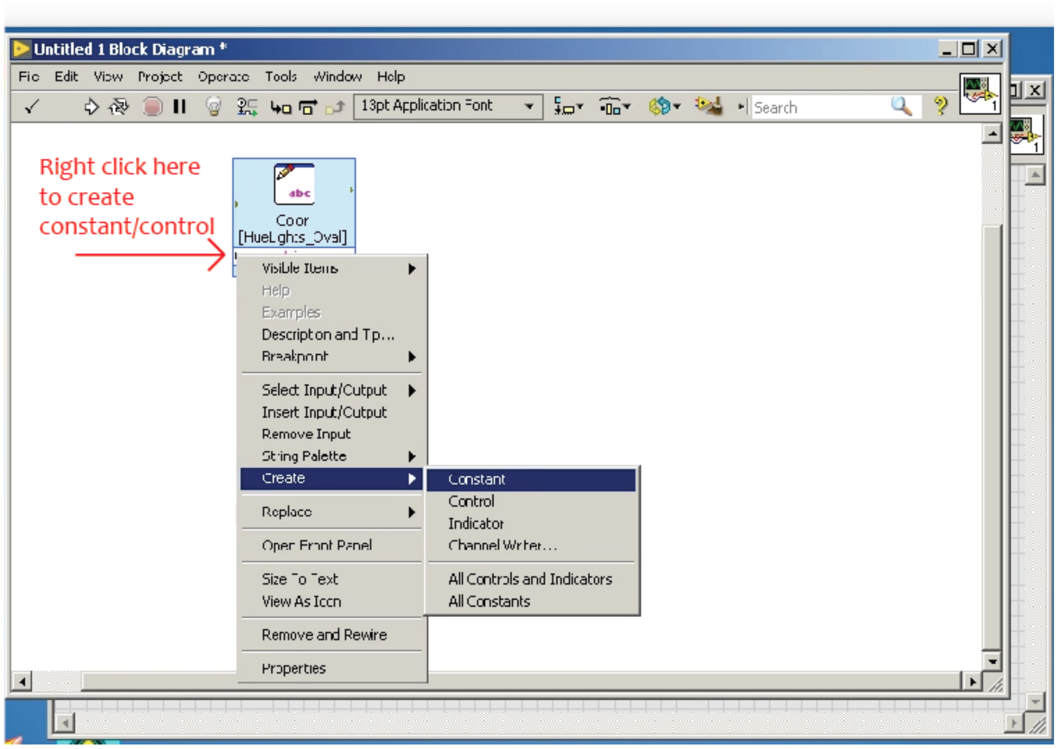
Press Control + E to navigate to the front panel/block diagram

How to get to the Toolkit Express VI's:

Right click an open space on your block diagram (white background)>IoT Education Toolkit> ThingWorx> Read/Write blocks

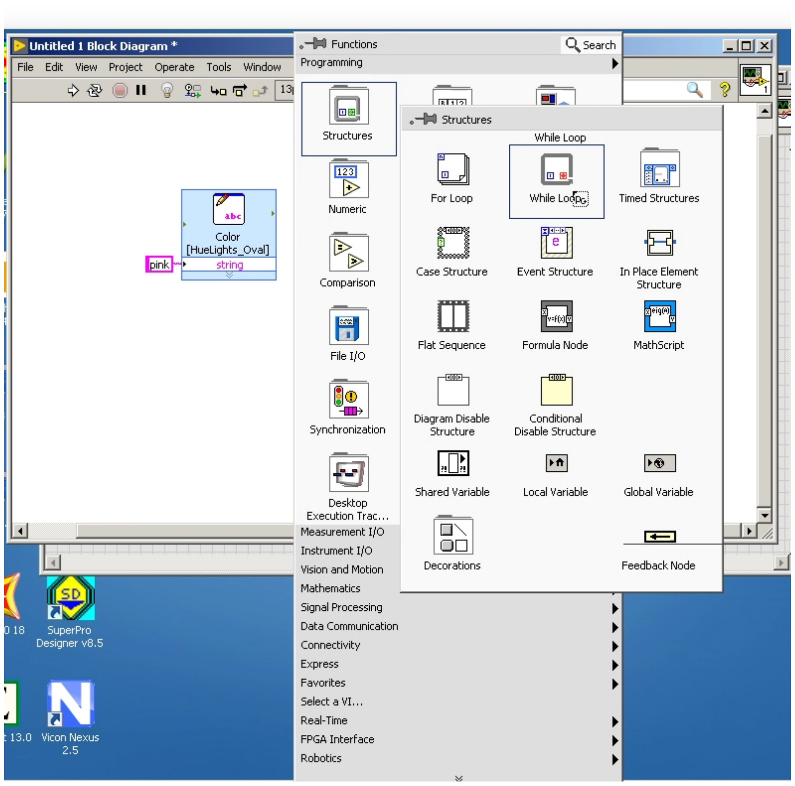


How to create constants/controls/indicators

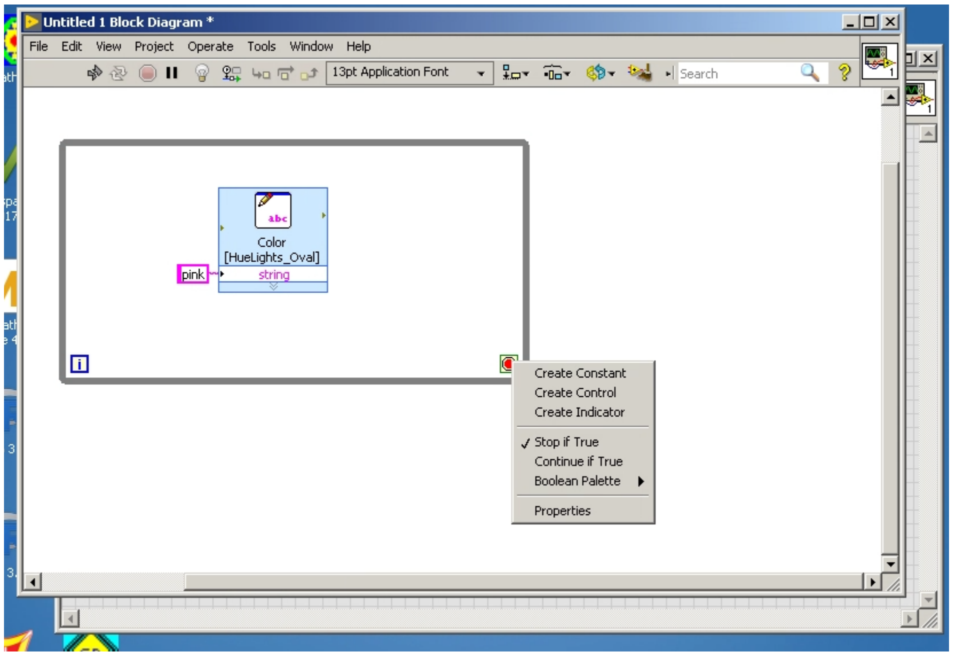


How to create a while loop

Right click on the block diagram (on a blank space) > structures > while loop

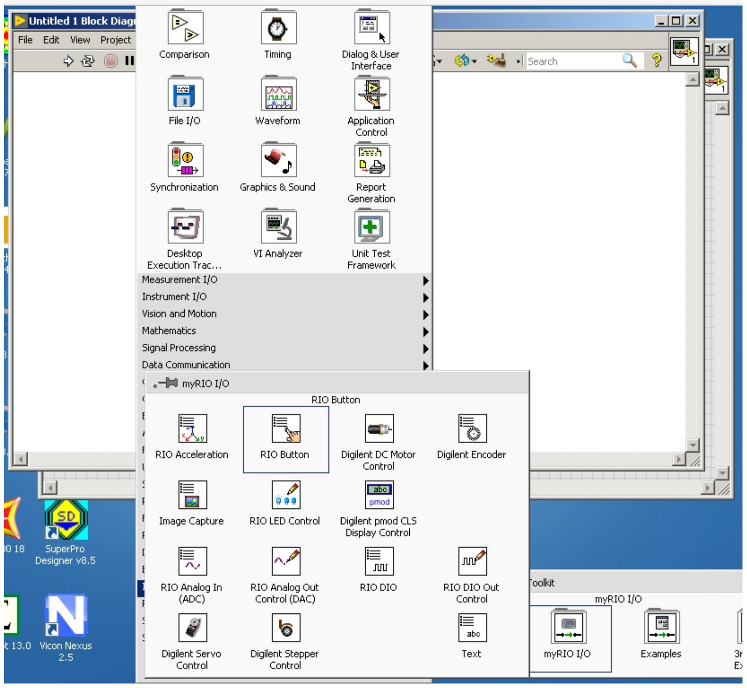


Make sure you right click on the stop icon and create a control



Where to find the myRIO button block in LabVIEW

Iot Education Toolkit > myRIO I/O > RIO Button



Appendix D - Usability Test Individual Results

Table D.1: User UP1

LabVIEW experience: Yes

Task number	Task completed	Time to complete task (min)	Correct post-task assignment diagram
1	1	14:18	1
2	1	7:07	1
3	1	10:27	1
4	0	-	1
Total	3	31:52	4/4 avg time (10:37)

Table D.2: User UP2

LabVIEW experience: No

Task number	Task completed	Time to complete task (min)	Correct post-task assignment diagram
1	1	16:20	0
2	1	17:30	0
3	0.5	08:06(i)	0
4	0	-	1
Total	2.5	33:50	¼ avg time(13:32)

Table D.3: User UP3

LabVIEW experience: Yes

Task number	Task completed	Time to complete task (min)	Correct post-task assignment diagram
1	1	27:52	1
2	1	13:02	1
3	0.5	15:33 (i)	1
4	0	-	1
Total	2.5	56:27	4/4 (avg time 22:35)

Table D.4: User UP4

LabVIEW experience: No

Task number	Task completed	Time to complete task (min)	Correct post-task assignment diagram
1	1	17:30	1
2	1	28:24	0
3	-	-	-
4	-	-	-
Total	2	45:54	½ avg time (22:57)

Table D.5: User UP5

LabVIEW experience: Yes

Task number	Task completed	Time to complete task (min)	Correct post-task assignment diagram
1	1	~5	1

2	1	~5	1
3	0.5	~5	1
4	-	-	1
Total	2.5	~15	4/4 (avg mins)

Table D.6: User UP6

LabVIEW experience: No

Task number	Task completed	Time to complete task (min)	Correct post-task assignment diagram
1	1	17:24	1
2	1	17:00	0
3	-	-	-
4	-	-	-
Total	2	34:24	½ avg (17:12)

Table D.7: User UP7

LabVIEW experience: No

Task number	Task completed	Time to complete task (min)	Correct post-task assignment diagram
1	1	14:12	0
2	1	20:37	0
3	0.5	11:03	0
4	-	-	-

Total	2.5	45:52	0 avg (15:17)

Table D.8: User UP8

LabVIEW experience: Yes

Task number	Task completed	Time to complete task (min)	Correct post-task assignment diagram
1	1	~16:00	1
2	1	~10:00	0
3	0.5	~8:00	0
4	-	-	-
Total	2.5	34:00	1 (avg 11:20)

Table D.9: Total Usability Participant Data

LabVIEW experience	Yes	No	Total
# of participants	4	4	8
Average # of tasks completed	2.63	2.25	2.44
Average time to complete tasks (min)	~14:53	17:15	~16:04
Average # of correct post-task diagrams	13/15	3/11	16/26

Appendix E - Interview Full Transcript for Selected Questions

PTC Developer

Question A

Did you see any clear limitations on the Toolkit as they relate to allowing students to add complexity?

Response A

“The cloud aspect is difficult. It's one of the most difficult parts of IoT, largely because there's no standard...If I’m getting data from Wunderground (a data source for weather information), it's way different than how I might get it from weather.com. Those two different platforms giving me pretty much the same information, but have no standard interface or format to them, so there's no good way to provide a student with a way to generically add new sources of data because there's no shortage of those there's hundreds of different sources of data. There's baseball data, there’s weather data, there’s geological data. So all of those exist, but there's no standard interface yet for those Things to be pulled in…

That concept extends to Things like analytics and to Things like machine learning. Some of those more advanced concepts, because there's no way to directly pull some of those engines and tools in to work on your data, because there's no standard interface for those either.

In the commercial space that's what ThingWorx does best is becoming that standard interface for all these different platforms to interact with each other. the issue is, is to open up the complexity of that, you just have to kind of be a fully trained software engineer which oftentimes in the time crunch that professors have in the University space isn't an option to get student fully trained and then do a project it's usually you train as you go and hopefully by the end have a good project, which I think the NI Toolkit accomplices much better than if you were trying to train someone on just ThingWorx or on just LabVIEW to do either of those.”

Question B

How did you hope that students would use Composer, if at all?

Response B

“For Composer the hope was that a student would eventually get to that point. The key aspect of ThingWorx Composer is that it is the full capability of ThingWorx or nearly the full in the academic version that's provided, which means you can do pretty much anything so you can integrate other data sources… you can do some more advanced analytics, you can send that data to another platform ... you can do all of these really complicated operations and have all that information be orchestrated. That's the key aspect of ThingWorx is that it’s data orchestration at its core, and the hope was that eventually students may get there. Especially if they have a very specific use case. They have the ability to meet that use case but it has to be done with a more complex tool.

That was a hope. I'm not sure how many students have gone from a simple use case that they were using the Toolkit for, and then extended on to that next step, but where it fit was definitely as a more advanced and intricate version of the cloud aspects of the Toolkit.”

NI Interview Full Transcript for Selected Questions

Question A

Do you see any clear limitations on the Toolkit as they relate to allowing students add complexity?

“Certainly being able to take the the myRIO I/O and customize what parts of that you want to write to the cloud to write to ThingWorx and vice-versa. If there is some data that's on the cloud that you want to be able to reflect on this edge node, the myRIO, the IoT Toolkit is a good way to do it. You need some code running on that target and there might be a better way to program that in the future but today the only way to do that is to either write a pre-configured image that's reading and writing all of the data and sort of to deal with the latency there or to write some custom code using this Toolkit to deploy it from LabVIEW onto the myRIO and now it's capable of talking to these cloud services all right you're saying this is.

...If the goal is to make a myRIO a cloud connected device and then you need something like this. If the goal is to inexpensively measure temperature of homes, [it’s] a terrible idea.”

Question B

How did you think the myRIO fits as a microcontroller that pairs with the Toolkit?

“For these types of projects again when the end goal is to read on an analogue you know an analogue measurement and you’re content with ten bits or twelve bits of accuracy on an analog sensor reading or you need to blink an LED or you need to you know turn on a relay or Things like that you don't need any of the power of an FPGA processor, 32 bits with 512 mega RAM and all of this other horsepower and all the protection of the I/O and all the other Things that the myRIO brings to the table for student applications, at $2.50 Arduino has more than enough horsepower for those applications and some of these platforms do have have Wi-Fi connectivity and have the ability to do those Things.

Now what they don't have is a tremendous programming experience, some of them, but you know connected to the Arduino ecosystem, if your application is relatively straightforward, well the the corresponding text-based Arduino codes it's also pretty straightforward. And so you know if you're competing against that, and [in this IoT

Toolkit] you're bringing this sort of heavyweight solution in order to Blink an LED, you're bringing the wrong set of equipment to the fight.

Appendix F - Student Interview Questions

Table E.1: Student Interview Questions and Responses

Question	Response
Did you use ThingWorx Composer at all? If so, how?	RS2: Talking about the ThingWorx Composer site. “That’s the one that ended up doing what we wanted it to do, and I think we also went to the dashboard, which didn’t do what we wanted it to do, because we were using a phone app, which is not using [Professor Roger’s] LabVIEW VI’s, so we were using GET and PUT requests, and we wanted to check it without going through [Professor Roger’s] VI’s to see, did we create this object, does it have the values we think it should have?”
How do you think the myRIO fits as a microcontroller that pairs with the Toolkit?	RS2: “I enjoyed myRIO. I do robotics research through a different lab and I sort of wish I had a myRIO to control my robot now, just because of how many different motors and encoders I have. The scale of the myRIO seems really good for robotics applications, that being said it’s really expensive. With the actual Toolkit, there’s a bit of an annoyance using that in conjunction with the myRIO simply because there are some issues getting the App key on it, using [Professor Roger’s] VI’s.” RS1: The Thing about the myRIO that sets it apart is it’s more robust, I think than a lot of other processors, and it will hold a Wi-Fi connection once you get everything set up and it tends to - I rarely have issues with it crashing once everything is the way I want it to be, so I mean that certainly says a lot. And also the FPGA is- you can do high level stuff with that so it’s fun… so when we do anything with audio really, is best done on the FPGA ‘cause it’s just so fast, and you get rid of any of that lag, so Things like voice recognition and storing the voice data.” RS4: The myRIO worked much better than the microcontroller that was needed for any of our projects. I definitely don’t think we needed a $700 microcontroller for a lot of these projects. It’s much more powerful than we needed. I know this is specifically because of the FPGA system on the microcontroller, and we did get to play around with that a bit, so we got to see some of its uses, but it might have been useful to play around with more Raspberry Pi systems that could be more lightweight and more suitable, so in a professional setting we would pick something more on that side than a full RIO, but for educational purposes, it was great to have the capabilities that the RIO does have.

	RS3: “I feel like they work very well together it’s very straightforward… It’s very user-friendly.” RS3 continued: “I feel like there’s a lot more inputs/outputs into the myRIO, something like Arduino you’re limited to about 10 inputs/outputs. The myRIO as built in accelerometer, a couple LED lights. The only issue with the myRIO is that it’s bulky. Also it has that FPGA, so that was also a pretty cool Thing to learn about.” RS5: The fact that the Toolkit is written in LabVIEW and myRIO has LabVIEW of its native language, probably cause it’s an NI Thing, that makes it really easy to understand that they go together… As far as the RIO being used as the processor- It wasn’t as important as far as the processor goes, so like if it was an Arduino, I would be super fine with that, so long as both the Toolkit and the Arduino were through the same language, so if you could get LabVIEW running on an Arduino, and could get the Toolkit working on that, I think then it would feel the same, except that it would be smaller than a RIO, and a little weaker.
Do you think there could have been another platform, application, or software that could have helped you learn this material better?	RS1 “I’m not sure I would’ve used something different, but I think there’s value in seeing lots of different Things. So ThingWorx was sort of the only cloud platform that we saw, and I think something that’s really essential to the whole “cloud conversation” is that there is Amazing Web Services, and there is the Google Cloud, and how all of these Things talk to each other is interesting…” RS1 continued “So like, why do I care- What made the myRIO and ThingWorx better than just an Ev3? Because a lot of times we were doing the same Things, like I was building a robot with the myRIO that I really could’ve done with an EV3 talking Bluetooth, and so trying to understand- Have that carrot of ‘why do I want to be on the Internet,’ and why do I want to be accessible to other people who are on the Internet?”
What was the goal of the course? In other words, what did you hope to learn?	RS1: “So I think the goal of the course was to go a step beyond sort of traditional robotics classes, which are building robotics for really clearly defined tasks, and this was supposed to be more of an in-depth look at how you can use robotics for more complex tasks, including robots that talk to each other, and that sort of have more intelligence, so it was not just focused on physically building robots, but also adding that intelligence into them”

What were the parameters of the class, as you understood them, as they relate to the actual IoT Education Toolkit?	RS1: “One of the parts of the syllabus was just the marriage between IoT and robotics, and so the Toolkit sort of was our means to connect our robots to the Internet. So we used it as a way to make projects that had some sort of- had more intelligence than what we could just do locally. They were able to pull data from the Internet. Once we had that sort of data it was how we incorporated it into our projects”
Are there features of the Toolkit or myRIO that you wish you had?	RS1 “Yeah, I wish the myRIO worked with a Mac, and I wish that it was easier to debug in general, like.. It’s a black box. Everything about it is just rectangular and closed off... Everything about it is tricky, like the Particle Photon has a really great app where you open it up and you can turn on digital pins and just really have like a baseline way to communicate with it... The myRIO has some Express VI’s where you can test Things, but it’s just difficult and I wish there was a website I could go to and I could turn pins on and off and I could just see what was happening with my device.” RS4: One Thing that I found difficult... to connect to third party Internet services from the RIO remotely, so when I was connecting the RIO to ThingWorx or another third party, I almost always did it through the computer, and the computer would pull the information, and through shared variables or one way or another, I’d get it to the RIO. I had difficulty setting up the system on the RIO Wi-Fi, although this may be because of the Tufts network. I’m not sure. It always causes problems one way or another.” RS5: On a myRIO it can be weird to tell what Wi-Fi it is connecting to. I don’t know how to check that besides typing into the IP [address] and looking at the… myRIO [Internet] monitor. I wish I had more control how myRIO connects to Things.
Did the Toolkit have any limitations that prohibited you from doing something you wanted to do on a project or learn in the	RS4: The biggest limitations we found is kind of the bandwidth that we could pass through ThingWorx, so if we had a lot of information being updated quickly, then we run the risk of missing some of it in the process of trying to receive it on the other end, so we didn’t find this to be too severe of a problem, usually, if we spent some time for a work-around, so an example would be if you wanted to create a proportional control, you wouldn’t want your position data being passed through ThingWorx and then coming back down and trying to calibrate

class?	because the delay would cause it to never quickly match what your goal was for the controller. RS5: “The biggest limitation that people had was the frequency in which ThingWorx gets information...ThingWorx has a half second to one-second delay sometimes, and when you’re controlling a robot, that can be really problematic, so that was a limitation, especially with image processing.”

Bibliography

Amazon Web Services, Inc. (2018). *Cloud Computing Services*. Retrieved February 20, 2018, from Amazon Web Services (AWS): https://aws.amazon.com/
Amazon.com, Inc. (2018). *Echo & Alexa Devices*. Retrieved April 2018, from Amazon: https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/b?ie=UTF8&node=9818047011
Brown, E. (2016, September 13). *"Who Needs the Internet of Things?"* . Retrieved March 2018, from Linux.com.
Callaghan, V. (2012). Buzz-Boarding; Practical Support for Teaching Computing Based on the Internet-of-Things. *1st Annual Conference on the Aiming for Excellence in STEM Learning and Teaching* (pp. 1-5). Colchester: University of Essex.
Carlson, J. (2010). *Gap Analysis-The Foundation of Customer Satisfaction Research.* Atlanta: Polaris Marketing Research, Inc.
Evan, D. (2011, April). The Internet of Things: How the Next Evolution of the Internet Is Changing EveryThing. *Cisco White Paper* .
Google. (2018). *Google Cloud Computing, Hosting Services & APIs*. Retrieved February 2018, from Google Cloud: https://cloud.google.com/
Gubbia J, B. R. (2013). Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Generation Computer Systems-The International Journal of eScience , 29* (7), 1645–1660.
Hackster.io. (2018, March). *IoT Made Easy: Capturing Remote Weather Data*. (MJRoBot, Producer) Retrieved April 2018, from Hackster.io: https://www.hackster.io/mjrobot/iot-made-easy-capturing-remote-weather-data-cb4b41
IFTTT. (2018). *IFTTT helps your apps and devices work together*. Retrieved February 2018, from IFTTT: https://ifttt.com/
Lawson, R. (2006). The Science of Cycology: Failures to Understand How Everyday Objects Work. *Memory & Cognition , 34* (8), 1667-1675.
Lohs, I. (2017, December 17). *MyFeeder: Remote semi-automatic feeding machine using Particle Photon & Blynk.* Retrieved 14 2018, April, from Hackster.io: https://www.hackster.io/ingo-lohs/myfeeder-a90455
Marra, M. B. (2018). A Gap Analysis Methodology for Product Lifecycle Management Assessment. *IEEE Transactions on Engineering Management , 65* (1), 155-167.
Microsoft. (2018). *Microsoft Azure Cloud Computing Platform & Services*. Retrieved February 2018, from Microsoft Azure: https://azure.microsoft.com/en-us/
National Instruments. (2018). *A Global Leader in Test, Measurement, and Control Solutions*. Retrieved March 1, 2018, from National Instruments: http://www.ni.com/en-us.html
National Instruments. (2018). *myRIO Student Embedded Device.* Retrieved April 2018, from National Instruments: https://www.ni.com/en-us/shop/select/myrio-student-embedded-device

National Instruments. (2017, June 14). *NI and PTC Collaborate to Bring IoT Education to the Engineering Classroom*. Retrieved March 1, 2018, from National Intruments: http://www.ni.com/newsroom/release/ni-and-ptc-collaborate-to-bring-iot-education-to-the-engineering-classroom/en/

National Intruments. (2018). *What Is LabVIEW?* Retrieved February 2018, from National Intruments: ni.com/en-us/shop/labview.html

Nest Labs. (2018). *Nest Thermostats*. Retrieved April 2018, from Nest: https://nest.com/thermostats/

Particle. (2018). *Welcome to IoT*. Retrieved February 2018, from Particle: https://www.particle.io/

Philips Lighting Holding B.V. (2018). *Wireless and Smart Lighting by Philips*. Retrieved April 2018, from Meet Hue: https://www2.meethue.com/en-us

Porter, M. E. (2014). How Smart, Connected Products Are Transforming Companies. *Harvard Business Review , 114*, 96-112.

Porter, M. E. (2014). How Smart, Connected Products Are Transforming Competition. *Harvard Business Review , 92*, 11-64.

PTC. (2018). *Technology Platforms and Solutions to Unlock the Value of the IoT*. Retrieved March 1, 2018, from PTC: https://www.ptc.com/

PTC. (2018). *ThingWorx Delivers Industrial Innovation*. Retrieved February 20, 2018, from ThingWorx: https://www.thingworx.com/

Stanton, N. A. (2017). Data Collection Methods. In *Human Factors Methods: A Practical Guide for Engineering and Design* (Vol. 2, pp. 14-19). London: Taylor & Francis Ltd.

The MathWorks, Inc. (2018). *IoT Analytics - ThingSpeak Internet of Things.* Retrieved March 2018, from ThingSpeak: https://thingspeak.com/

The MathWorks, Inc. (2018). *MATLAB Licensing for Campus-Wide Use*. Retrieved April 2018, from MathWorks: https://www.mathworks.com/academia/matlab-campus.html