Robotics Engineering

Steve Cline

2020-07-28

Contents

‘‘‘ options(tinytex.verbose = TRUE)

Chapter 1

Prerequisites

While we will use Canvas as much as possible in this course, it is not a suitable platform for a course which relies heavily on programming. As a result, you will need to become familiar with using sites like GitHub. Familiarity with these sites will prepare you for future work in college and beyond.

For this course you will need to do the following to prepare to learn:

Locate the Course Lecture Notes section of this site. Browse through the notes to gain an understanding of what we will be covering this year.

Sign up for a GitHub account and fill out this form.

Bookmark the course GitHub Repository. This is where many of the programming resources will be found. In addition, assignments will be submitted here.

Locate our two textbooks for the course. They are both online and free. The first is an electronics textbook called Lessons in Electric Circuits. The second is a programming textbook called Think Python.

Sign up for an OnShape account by following the instructions in the OnShape Account Setup section of this site.

Complete the WHS Mechatronics Lab Code of Conduct Agreement.

Chapter 2

Introduction

2.1 A brief Overview of the Course

Remember that engineers spend a majority of their time debugging their designs. You will too. It is not because this course is made too hard for you...it is just the way engineering works.

"They say that no plan survives first contact with implementation. I'd have to agree." – Mark Watney, Sol 40

Why begin with this quote? One of my favorite books and movies of all time is The Martian. The struggles of the main character, Mark Whatney, demonstrate the challenges facing engineers. While we will not be risking our lives on another planet, our solutions to the problems in this course, follow the same pattern as his. Where he sought to survive being the only human on another planet by engineering his way out, we will survive this course by learning to attack problems in the same way and remember not to give up. We must keep in mind, no plan will work the first time.

During this class students will explore the field of mechatronics using a variety of hands-on activities. Students begin the semester with an introduction to basic history and theory of robotics, the engineering process and tools and processes used to create robotic devices. We will introduce basic electronics concepts. Moving forward programming becomes an essential and vital element. Students program the onboard micro-processor found on a Raspberry Pi 3B+. This control board will use the Raspian OS which is a version of Linux. While students will work within the Linux shell, the programming language of this course is Python. Students work individually and in teams to design and build simple and complex mechatronic systems capable of meeting a variety of criteria including driving, pushing, controlling speed, etc. Sensors are introduced to allow robotic devices to interact with the environment. Actuator design is dis-

cussed and different manipulator systems are introduced. As an essential part of mechanical design, students will be exposed to CAD concepts using OnShape.

As you can see from this Euler diagram here, Mechatronics is a complicated field of engineering that combines many areas of study. This is the more technical name for robotics and was created by Tetsuro Mori in 1971 and has served as the name for this field since. In this course, you will be introduced to several of the areas on this diagram.

Since much of this course will be delivered remotely, be sure that you consider creating a good remote learning environment by doing the following:

Create a dedicated workspace for yourself that you can keep neat and organized throughout the year. Pick a place that is as quiet as possible so that you can concentrate. Do not use this space for anything else. In this course that space should include a place for your computer as well as an area to work on electronics projects - preferable with a hard surface like a desktop or workbench.

Manage your sleep schedule. You will need to be able to concentrate and stay focused since you do not have a teacher hovering over your workspace to keep you focused. If you go to bed at a consistent time and wake up at a consistent time, you will be able to learn more effectively.

Make sure you know how to log on to every account needed for the course. See the Prerequisites for all of the details for this course. Be sure to ask questions if you encounter any difficulties.

2.2 History of Robotics

Adapted from Robotics: A Brief History, Stanford University

Origins of “Robot” and “Robotics”

The word “robot” conjures up a variety of images, from R2D2 and C3PO of Star Wars fame; to human-like machines that exist to serve their creators (perhaps in the form of the cooking and cleaning Rosie in the popular cartoon series the Jetsons); to the Rover Sojourner, which explored the Martian landscape as part of the Mars Pathfinder mission. Some people may alternatively perceive robots as dangerous technological ventures that will someday lead to the demise of the human race, either by outsmarting or outmuscling us and taking over the world, or by turning us into completely technology-dependent beings who passively sit by and program robots to do all of our work. In fact, the first use of the word “robot” occurred in a play about mechanical men that are built to work on factory assembly lines and that rebel against their human masters. These machines in R.U.R. (Rossum’s Universal Robots), written by Czech playwright Karl Capek in 1921, got their name from the Czech word for slave.

The word “robotics” was also coined by a writer. Russian-born American science-fiction writer Isaac Asimov first used the word in 1942 in his short story

"Runabout." Asimov had a much brighter and more optimistic opinion of the robot's role in human society than did Capek. He generally characterized the robots in his short stories as helpful servants of man and viewed robots as "a better, cleaner race." Asimov also proposed three "Laws of Robotics" that his robots, as well as sci-fi robotic characters of many other stories, followed:

A robot may not injure a human being or, through inaction, allow a human being to come to harm.

A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.

A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Early Conceptions of Robots

One of the first instances of a mechanical device built to regularly carry out a particular physical task occurred around 3000 B.C.: Egyptian water clocks used human figurines to strike the hour bells. In 400 B.C., Archytus of Taremtum, inventor of the pulley and the screw, also invented a wooden pigeon that could fly. Hydraulically-operated statues that could speak, gesture, and prophecy were commonly constructed in Hellenic Egypt during the second century B.C.

In the first century A.D., Petronius Arbiter made a doll that could move like a human being. Giovanni Torriani created a wooden robot that could fetch the Emperor's daily bread from the store in 1557. Robotic inventions reached a relative peak (before the 20th century) in the 1700s; countless ingenius, yet impractical, automata (i.e. robots) were created during this time period. The 19th century was also filled with new robotic creations, such as a talking doll by Edison and a steam-powered robot by Canadians. Although these inventions throughout history may have planted the first seeds of inspiration for the modern robot, the scientific progress made in the 20th century in the field of robotics surpass previous advancements a thousandfold.

The First Programmable System

There is very little doubt today about both how essential programming is to robotics and who the earliest computer programmer was. While many would immediately think of modern computer scientists like Steve Wozniak or earlier programmers like Alan Turing, the first person to successfully put forth the idea of a programmable system like a computer was Ada Lovelace. In her article of 1842 (Yes, during the John Tyler administration) she proposed that numbers and objects could be used to "express abstract scientific operations" and create action based on rules. This was the first time anyone had proposed programming a device to act on inputs based on rules. This is the very heart of what it is to be a robot. Despite the great disadvantage of being born a woman in a very male dominated 19th century, she founded the field which would become the backbone of mechatronics.

The Origins of the Math of Robotics

The backbone of decision making in the mechatronics is a field of math called Boolean Algebra. As a professor of mathematics at Queen's College Cork in Ireland, George Boole published a work titled An Investigation of the Laws of Thought in 1854. Boole seeks to prove mathematically the existence of god through the application of logic. While his success at justifying the existence of god has not been supported, the underlying mathematical system that he invented would be used a century later as the foundation for modern day computers.
Without Boole's contribution to mathematics and logic, modern microprocessers built from transistors would not be possible. While he lived in obscurity during his life, the long legacy of George Boole is found in every computer programming language, including Python. The code below, which you will become familiar with this year, demonstrates this application.

```
While True:
      print("Boolean Algebra makes computer decision making possible.")
```

For a more in-depth understanding of his contributions to mechatronics see the embedded video in the Supplemental Videos Section.

The First Modern Robots

The earliest robots as we know them were created in the early 1950s by George C. Devol, an inventor from Louisville, Kentucky. He invented and patented a reprogrammable manipulator called "Unimate," from "Universal Automation." For the next decade, he attempted to sell his product in the industry, but did not succeed. In the late 1960s, businessman/engineer Joseph Engleberger acquired Devol's robot patent and was able to modify it into an industrial robot and form a company called Unimation to produce and market the robots. For his efforts and successes, Engleberger is known in the industry as "the Father of Robotics." Academia also made much progress in the creation new robots. In 1958 at the Stanford Research Institute, Charles Rosen led a research team in developing a robot called "Shakey." Shakey was far more advanced than the original Unimate, which was designed for specialized, industrial applications. Shakey could wheel around the room, observe the scene with his television "eyes," move across unfamiliar surroundings, and to a certain degree, respond to his environment. He was given his name because of his wobbly and clattering movements. The Supplemental Videos section of this text has a video made about Shakey.

In order to better understand the field of mechatronics, you will choose one person who has been foundational to the development of the field.

Assignment: The Innovators Project

Chapter 3

Electrical Concepts

This class and modern life relies heavily on electricity and electronics. Our course electronics text points this out quite eloquently:

The history of electricity starts more than two thousand years ago, with the Greek philosopher Thales being the earliest known researcher into electricity. But it was Alessandro Volta who created the most common DC power source, the battery (for this invention the unit Volt was named after him).

Direct current (also known as DC) is the flow of charged particles in one unchanging direction (most commonly found as electron flow through conductive materials). DC can be found in just about every home and electronic device, as it is more practical (compared to AC from power stations) for many consumer devices. Just a few of the places where you can find direct current are batteries, phones, computers, cars, TVs, calculators, and even lightning.

We will begin exploring a number of concepts relating to electricity as we will essentially be using electrons as our modes of information processing. From the electrons that flow through a button or distance sensor to be read by the microprocessor to those which supply the power for our motors and servos, we are heavily dependent on electricity. Before we get into this, we need to go over some very important concepts. Take out your Raspberry Pi and follow along with the video below.

Video here Rules of this course in regards to electricity:

Always work only with DC electricity, never use AC voltage except for when you plug your Raspberry Pi into the outlet.

Always work on your circuits when the devices are unplugged and batteries removed. Not only is this a safety issue, it will prevent the need to replace electrical components

Only use the components listed in the laboratory activity on which you are working. Do not try new components you have never used before or are unfamiliar with.

Keep your work area clean and uncluttered. Do not keep beverages on your workstation.

Using EasyEDA for circuit design

There are many useful tools available to help with circuit design. During this course we will need to learn to use some of them. Circuit schematics make it easy for us to design and interpret circuits to communicate our project. For this class we will use EasyEDA. We can run it in a web browser. It allows us to save projects to a free account and share them for submission. If you learn how to use it well enough you may also use it to design your own printed circuit boards (PCB) to make your own components.

Go to the EasyEDA editor website. Create an account. While you may be able to just start using it and learn through trial and error, here are a few tutorials that will become useful which may be found in the Tutorials section of the app:

Introduction to EasyEDA

UI Introduction

Create a New Project or File

Schematic Capture

Basic Skills

You should spend some time learning how to use this app. We will begin using it soon and you will learn how to read an electronics schematic, by designing them yourself. Once you have set up your account and learned some basics, complete the following assignment:

Assignment 3.0

Using your newly discovered schematic design skills, you will make your own basic schematic. Watch this short tutorial explaining schematics. Create and save a schematic in EasyEDA with the following components in a circuit:

5V power supply

220 Ohm Resistor

Blue LED

Ground

3.1 Circuits

A basic understanding of circuit electricity is essential going forward. For this section of the course, we will begin using our online textbook. It gives us an excellent overview of electrons, circuits and polarity. Please keep in mind that, in order to begin using the Raspberry Pi (i.e. receiving the power supply) you will need to pass all of the required quizzes with at least a 70%.

Key Concepts of Section 3.1

A circuit is a loop of conductive material.

In order for electrons to flow a closed circuit is required.

A circuit is open or broken if a complete path for the electrons to flow no longer exists.

Rather than me trying to do a better job than our textbook, you are going to read the short section on circuits. These questions will be similar to those that you will see on your unit exam (The one you need at least a 70% on to move on). So…that means that you are not completing this worksheet for a grade. Instead, you are trying to measure how well you learned the concepts in the reading. Much of this year's content will be delivered this way. Get your head around the idea of learning for the sake of learning.

To help you understand this concept, here is an excellent video from SparkFun Electronics on the topic.

Assignment 3.1

After reading the section on circuits, complete the worksheet

Using EasyEDA, design a closed circuit with the following components:

5V power supply

220 Ohm Resistor

Blue LED

Ground

Submit the your work on GitHub through Assignment 3.1.

3.2 Voltage and Current

The movement of electrons from one pole to another is what we use in robotics to make our robots work. By altering the volume of electrons moving through the wires connected to a motor we can alter the speed of the wheel attached to the motor. The potential energy coming in to our microprocessor pin over a wire attached to a photoresistor tells us the level of brightness or lumosity in

a room. Putting these two concepts together we could create a robotic system that opens the blinds on our lab when the sun sets in the west. That is, of course, if we can program our system to make that decision on its own. We will get to the programming part later. For now, we will examine the concepts of voltage and current which allow us, along with resistance, to receive information through sensor and send information out to an actuation device like a motor.

Read short section on voltage and current. Be sure to review the concepts at the bottom of the page for understanding.

To help you understand this concept, here is an excellent video from SparkFun Electronics on the topic.

3.3 Resistance

The final concept needed to understand how electricity in a DC circuit may be used to control devices is resistance. One day last year I was sitting in my office upstairs. I had just come from the mechatronics lab and had packed a bunch of supplies from the lab into my backpack so I could do a little work on a demonstration I would be giving later in the week. All of a sudden I smelled smoke. I looked down and it was coming from my backpack! After digging around a bit I found the culprit. I had put a battery pack in my bag like the one here:

Can you see the problem? I had left all of the batteries in the pack. When I threw it in my bag, the red and black leads touched and starte what is know as a short circuit. With no resistance on the flow of electrons through the wires. The by product of this rapid flow was heat which began to melt the inside of my backpack and start it smoking. Remember this example and always pull one battery out of your battery packs when not connected to your device. Also, never connect two poles of a power source without any resistance. It can cause serious injury.

This problem is prevented by placing a resistive force on the circuit. Small electrical components like LEDs and sensors cannot handle even the two amperes of current available from the Raspberry Pi's 5V pin. A reisistor is used to prevent the LED from burning up. In order to understand how resistance works, read short section on resistance.

Assignment 3.3

After reading the section on circuits, complete the worksheet

3.4 Ohm's Law

All three of these concepts come together in Ohm's Law. In the 19th century many scientists were experimenting with electricity in an attempt to understand it. One of these scientists, Georg Ohm, determined that there was a relationship between Voltage, Current and Resistance. This is why an LED burns up (receives too much current) when no resistance is placed on the circuit. If we create resistance on the circuit with a resistor the voltage, or potential energy, remains constant but the current decreases. Ohm's law is represented by this simple formula:

$$V = I * R$$

As we will see, this formula is quite handy. It may be rearranged to solve for any one of the three possible outputs (V-voltage, I-current, R-resistance). In order to understand how resistance works, read short section on Ohm's Law.

To help you understand this concept, here is an excellent video from SparkFun Electronics on the topic.

Assignment 3.4

After reading the section on circuits, complete the worksheet

3.5 Using a Digital Multimeter

This section will become active in the event that we are not using remote instruction.

3.6 Basic Soldering

This section will become active in the event that we are not using remote instruction.

Chapter 4

Mechanical Design

The inspiration for this course at Windsor High School is actually a course in the Engineering College at Colorado State Unversity called Mechatronics and Measurement Systems. That course is meant to be one of the capstone courses in the mechanical engineering degree at CSU. In order to complete our capstone project, we will all need to have a solid understanding of the parts of a mechatronic or robotic system.

4.1 Parts of a Robotic System

We need to be able to determine what components will be necessary in our robot design. To do this, an overview of the various part types is necessary. At the end of this chapter, you will create a design on OnShape which will serve as the basis for your capstone project next semester. Use this as an opportunity to start thinking about what you will do in creating a robotic system.

Steps to prepare for this work

Start thinking about a problem you might like to solve. Is there something at home that could be improved? For example, last year I built a solar powered Raspberry Pi controlled garden watering system. Something like that has a definite use and provides a solution for my wife and I. Now that it works, when we go on vacation we don't worry that our garden gets watered.

Look for other examples. A good place to start is the Student Examples from the MECH 307 Course at CSU. Another good source is SparkFun Projects Examples. In additon, there are many websites that have Raspberry Pi ideas.

Think about what makes you excited to do this project. What would you have fun doing for four months next semester?

Write down some ideas so that, you may come back to them later as you learn more

Now that we have some ideas, let's learn about the various types of parts available to us.

4.1.1 Power Supply

A major consideration is power. Most of the systems we design need to be efficient in their use of power. We are not always able to connect to an outlet. For example, home surveillance cameras are small and easy to install in many different locations. However, if we had to connect them to an AC power source, the list of possible locations for installation could be greatly reduced. The solution is a battery pack. How do we decide which battery though? A surveillance camera might need to run for weeks at a time without changing the battery. As a result, a high quality lithium ion source would probably make the most sense. The drawback to this compact and powerful design is, of course, price. These are the types of trade-offs we must consider.

Power Supplies include:

DC Wall adapters

Battery packs for rechargable NiMH batteries

Pre-built battery packs

Solar panels

As we see, there are many options and considerations. Only by determining total power needs, space and weight constraints and budget can we select the correct source. For now, consider power needs first. As you get closer to the start of your project you can narrow down your choices.

4.1.2 Actuators

Most robots move and interact with the physical world. We will not make that distinction here although there are many who believe that this is a requirement. A robotic car system has many actuation systems. The most notable of these is the drive system. Through the wheels and transmission the motor interacts with the surface the robot is on to propel the robot. Any device creating motion in this way is an actuator. Like power supplies, actuator selection meets many criteria. A look at the RobotShop Actuator Page reveals hundreds of possibilities.

DC Motors

Stepper Motors

Servos

Linear Slides

From these fours types of actuators, most robotic systems may develop a way of interacting with the environment.

4.1.3 Sensors

There are even more sensor types than there are actuator types. If we brows the Adafruit Sensor Page we see how limitless these possibilities are. For any type of sensor application there is, someone has though of a way to build it.

As excerpt from a robotics text explains, sensors are either proprioceptive or exteroceptive and, at the same time, either passive or active. This chart is taken from that text:

4.1.4 Microprocessors

For most students in this course, the Raspberry Pi will function as the microprocessor for the course. Students learn to interact with the Pi using the command line and Python. In addition, the Raspberry Pi offers many advantages over other microprocessors like Arduinos. One of the main advantages is the built in wifi capabilities. This allows for remote programming and true Internet of Things (IoT) capabilites. Still, some may choose a different processor, maybe even a different version of the Pi.

A Raspberry Pi Zero W is an extremely compact and low cost alternative to the Rapberry Pi 3 we use in class. It is made for low-power, compact IoT applicaitons and should be considered. While the processor speed is slower than larger Pis, it might be just the fit for your project. Compact versions of other processors are available as well. This list is quite extensive and should be considered.

4.1.5 Structural Parts

The skeleton of the robotic system determines the physical constraints on the design. For years now, our robotics team has dealth with this issue. Several years ago they were presented with a challenge in the competition which required their robot to climb a mountain:

As we can see from the photograph, the mountain was quite daunting. The previous year, the team had advanced to the world championships and found that their robot, while the best design in Colorado, was not suited to compete with the heavier robots beyond the state. As a result they had been pushed around a bit and not been very successful. The solution to this was to weld a

steel frame together and build the robot off of that. The result was a 70 pound robot! Without thinking more deeply about the design of the current year, they had hampered themselves by dealing with the previous yesr's challenge in mind. The robot was robust and strong but could not climb the mountain. Instead, every tread system they tried to use to climb was under so much strain that it eventually broke. They would even 3D print treads made out of high strength plastics that still could not hold up to the strain. In this case, the structural parts let them down. What a great learning experience that was. All three of those students in the photo are in science and engineering programs currently in college. The one on the right, Mitchell Watson, is literally going to be a rocket scientist when he graduates for Cal Tech. Understanding these concepts of mechanical design are not only fundamental to robotics, they can lead to many great opportunities.

There are many options for structural parts. Over the years, we have accumulated a wealth of parts in our program which may be checked out. In addition, we have plenty of scrap material (we hated that mountain so much that we cut it up at the end of the season and have been using it for scrap since then). After you learn how to use CAD in the next section, you will be able to design and print your own custom parts as well.

The type of material matters. Begin with this materials guide in selecting your parts. Consider using pre-built kits or components. Here is a list of companies that provide robotics parts:

Pitsco Education: Many different types of parts from a long-time vendor

REV Robotics: Higher quality aluminum and high grade plastic parts

GoBilda: Awesome parts for large applications

80/20 Inc.: Incredibly versitile structural systems

RobotShop: If you are looking for a kit, to get started

SparkFun: Based in Niwot, you can go visit them and pick up your parts or take the tour to get some ideas

NOTE: For any vendors, check before you buy. We have educational discounts with most all of them.

Assignment 4.2 On graph paper draw a design for your expected capstone project. You should include at least three different angles for your drawing (two sides and a top). You should include dimensions and labels of each part. On a Google spreadsheet, make a list of all parts you think you might need. In a one-page typed summary describe your design.

4.2 Computer Aided Design

Once you have your design down on paper, it is time to create it like a real mechatronics engineer. For this section you will learn to use OnShape, a modern and full-scale computer aided design (CAD) system. OnShape was founded by two engineers at SolidWorks which is still considered the gold standard of CAD. SolidWorks comes with a hefty price tag, even for students. OnShape is free to secondary schools. As a result, we will use it. Once year learn this tool, switching to other CAD software will be relatively painless. In addition, it is entirely web based so you can use it as long as you have an internet connection and a web browser.

If you have not done so already, head to the OnShape Account Setup section of the course book. Follow those instructions. After you have done this, log on to your account and follow these instructions:

Click on the "Learning Center Button"

Select "Self-Paced Courses > Learning Pathways"

Select "OnShape Fundamentals: CAD"

Assignment 4.2 Complete the first five courses in this learning pathway:

Navigating Onshape

Introduction to Sketching

Part Design Using Part Studios

Multi-Part Part Studios

Onshape Assemblies

Chapter 5

Python Programming

Finally! We are going to start learning how to make the Raspberry Pi control components like lights and motors and receive input from sensors. This is what will make your creations become real robots. For this unit we will refer to the course programming text quite often. Many of the programming problems come from the Downey text as well. For all of your assignments in this section you will start here though. Learn the basic concepts of that assignment, read more about it on the course text, go to GitHub Classroom to practice the program and finally, transfer what you have learned to the Raspberry Pi.

Let's get started by setting up the Pi and learning more about Python and it's wide range of applications.

5.1 Setting up the Raspberry Pi

These procedures are for students in the Robotics Engineering course at Windsor High School. As such, some of the initial steps required in setting up a Raspberry Pi have already been completed. If you are using a new, out of the box, Pi refer to one of the many tutorials online.

Equipment Needed:

Raspberry Pi 3B+

5V | 2.5A power supply

32GB Micro SD card with pre-installed Linux based OS

Monitor

HDMI cable

Keyboard

Mouse

Optional: Windows, iOS or Linux Computer (not a Chromebook)

Connection to a LAN. In the lab this is "Mechlab". At home it is your router.

NOTE: For the initial setup we will use the monitor, keyboard and mouse. Eventually, we will only use the command line functions which will allow you to eliminate those components and control the Pi from your laptop remotely if you choose. If not, you can use the Pi as a computer itself but just enter the command line controls of the Linux terminal. Eventually, you will need to be able to do this though as we will disconnect the Pi from all physical I/O devices and make it the brains of a self-sufficient robotic system.

5.1.1 Putting the Pi Together

To begin, assemble the Pi by putting the SD card in the slot. Connect the monitor to the HDMI port, and the keyboard and mouse to the USB ports. Plug in the power cord and turn it on.

(Create a custom background for the RPi and install all of the programs needed for the class.)

5.1.2 Controlling the Pi with SSH

For many, it may be easiest just to use the Raspberry Pi as a standalone computer. In order to do this one just needs to keep the Pi connected to a monitor, keyboard and mouse. One of the main advantages of this is that one can just turn on the Pi, wait for the boot sequence to finish and start programming using the terminal. Another advantage is that other programming interfaces may be used such as Notepad++ which has built in code highlighting and markup functions that make programming in Python, and other languages, easier. More advanced users might even choose to install an interface like PyCharm which has even more functions aimed at enhancing the programming experience and may be connected to Git or an online code repository like GitHub. Maybe the biggest advantage of programming from the Pi desktop environment is that a web browser may be easily used alongside the command line terminal or programming interface.

Eventually though, we will all need to sever our connection to the peripherals. Imagine trying to test out a mobile robot system with the monitor still attached. When we make the switch we start interacting with the Pi through SSH (Secure Shell). This concept is not as difficult to understand as one might think. Essentially you are accessing the terminal window for the Pi through another computer using a piece of software like Putty. When both devices are connected to the same router, this is a fairly simple process. While we will not require this yet, it is strongly recommended that every student go over this process which is

detailed in the section titled Controlling the Raspberry Pi Remotely in the last chapter of this book sometime in the first semester. The Capstone project will require this so we might as well get used to it now. Most will probably switch back and forth between using the Pi as a desktop computer and accessing it via SSH.

5.2 The parts of the RPi and Pinout

Let's examine the parts of this tiny computer that you will be using for the rest of the year...it is really quite amazing how much can be packed on the small device.

There are several things worth noting here about the Pi:

While it is possible to power the device with a micro USB power supply that provides only 1A of current (older Android phones use this), it is best to use the full 2-2.5A power supply, especially when we start hooking up components to the GPIO pins

Be careful not to short two GPIO pins together or connect an external DC power supply incorrectly to one of the GPIO pins. These actions can damage essential components of the Pi.

TBC

A powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. There are 40 pins which we can use for a wide range of robotics applications. It is important to remember which pins are which as the locations do not represent the GPIO labels. If you forget or cannot find this diagram, use the command line in the terminal.

This is as good a place as any to start using the terminal. Before the introduction of the Apple Lisa with a graphical user interface (GUI), all computers used something like the terminal as the interface with humans. All computers still contain a terminal. The Raspberry Pi OS we are using is based on the Linux kernel so we will use Linux commands but the procedure is similar for other systems such as Windows and macOS. With the Raspberry Pi powered on press CTRL+ALT+T to open a terminal session.

After the terminal opens, type the following command: `pinout`

You should see something like this in the terminal:

Any of the GPIO pins can be designated (in software) as an input or output pin and used for a wide range of purposes. We will use them to send and receive electrical signals. Two 5V pins and two 3V3 pins are present on the board, as well as a number of ground pins (0V), which are unconfigurable. The remaining pins are all general purpose 3V3 pins, meaning outputs are set to 3V3 and inputs are 3V3-tolerant.

A GPIO pin designated as an output pin can be set to high (3V3) or low (0V). In addition pins GPIO12, GPIO13, GPIO18, GPIO19 may also be used with Pulse Width Modulation (PWM). A GPIO pin designated as an input pin can be read as high (3V3) or low (0V). This is made easier with the use of internal pull-up or pull-down resistors. Pins GPIO2 and GPIO3 have fixed pull-up resistors, but for other pins this can be configured in software.

It is possible to control GPIO pins using a number of programming languages and tools. For this course we will use Bash, the scripting language of Linux, and Python.

NOTE: Before you start your first programming assignment realize that in order to progress in this course you must fully learn the information in the text. If you skim or rush through you might be able to get many answers on the chapter quiz correct but you will miss many of the points. Programming is hard and learning to program is as well. It does not mean it is impossible. It just requires work. Take your time and make sure you are understanding what you read. If not, do the following:

Re-read the section.

Ask someone else in the class for clarification.

Google your question

Ask your instructor

This is good advice for all courses, especially the further you go in school. Many of you will go so far in college that you basically stop taking classes and just learn on your own for the sake of understanding something.

Assignment 5.2 For this assignment you will learn some basic commands and functions in Linux and how to find out how to use them. Open up a terminal on your Raspberry Pi as before.

Type the following: `help`

This is a list of all the built in functions that come with this version of Linux. As you add more packages and create your own functions this list will get larger. If you scroll to the top your see this message: `Type 'help name' to find out more about the function 'name'.` Let's try it. One of the functions you will use fairy often is `cd`.

Type `help cd` and answer the following questions from the information on the screen:

In your own words that you understand, what does this function do? (You may need to Google some of the terms like "directory" to understand what that term means in Linux)

If you type `cd Desktop` what do you think will happen?

Now call up the help information for the function `if`. Why would we ever use this function? Again, you might need to do a little research to understand the vocabulary here.

Write down or type your answers so we can discuss them in class next time.

5.3 Why Python?

It is worth spending a little time discussing the reasons why Python will be the chosen operating system for this course. Prior to Fall 2020, the course utilized the Arduino UNO as the microprocessor for the robotics systems. The Arduino is also a versatile, fully programmable microcomputer with input and output pins capable of producing and detecting low DC voltages. It utilizes the C++ programming language. There were a few limitations, however, with the Arduino UNO.

The biggest advantage of the Pi is the programming interface. Since the Pi is essentially a small Linux computer, applications and environments that function on Linux also function on the Pi. The default scripting language on Linux is called Bash. When you called up the `pinout` for the Raspberry Pi earlier, you were using Bash. It is a powerful command line interface. Why not just use Bash instead of learning another language within the Linux shell? The main reason for this is Python's usefulness outside of this class. We know from past experience that almost all of our Robotics Engineering students will go on to some kind of technical field of study in college. While those do not all go in to robotics, they all need to learn new languages. Python is about the most useful language one can learn in high school if planning to go on to a technical degree in college like engineering, computer science, data science or machine learning as well as almost any natural science field of study. It is powerful, easy to learn and widely used. The most recent StackOverflow survey ranked Python 3rd most popular among users. By learning this language, you will be more prepared to take on the challenges of a field of study in college related to robotics. Since one of the main objectives of our program is to help students prepare for post-secondary STEM fields of study, Python is an easy choice.

Python is also an interpreted language instead of a compiled language. What is the difference? In order to get a compiled language to run on a computer, the additional step of compiling it into machine code must take place before the device will run the commands of the program. With an interpreted language, a program or virtual machine running on the device (In this case, the Python environment installed on the Raspberry Pi OS) interprets the script written by the programmer directly into machine code. This is a faster process than running a compiled language like C++. Other interpreted languages include Ruby, Java and Perl and good old BASIC (Which many of us adult programmers learned when we were younger on our Commodore 64 or Apple IIe computers). This structure makes Python faster to learn.

There are other advantages to the switch to the Pi. The first advantage that the Pi has over the UNO is an onboard wireless internet connection. The Pi can connect to other devices using either 2.4GHz or 5GHz frequencies as well as Bluetooth. The current version also includes a Power over Ethernet (PoE) capability. Making the switch to the Raspberry Pi will allow for more seamless IOT (Internet of Things) activities without the need for an additional board. Second, the Raspberry Pi can actually function like a computer. We can modify the OS easily by modifying the micro SD card used as RAM on the computer.

5.4 Hello, World...and a Little More

It is a tradition in in programming to test out one's ability to program in any language by creating a program called "Hello, World!". This dates back to 1974 at Bell Labs. The actual C language code that was written was:

```
main( ) {         printf("hello, world\n"); }
```

In our case, we will do something very similiar. The first program we write will do the same as that one written in 1974. Open a terminal window by pressing CTRL+ALT+T. This will open a window that looks like this:

In order to run a command in Python through the terminal we need to set up the Python environment. Type this command into the terminal:

```
python3
```

This will show a message like this:

You know that you are able use Python commands when you get the `>>>` prompt. This is universal for all Python editors and the specific prompt for Python. We will try two methods of the "Hello, World!" program. First, we will use the print command. Type the following command into the interpreter:

```
print ("Hello, World!")
```

The quotation marks signify that we want to literally display what is directed by the `print` statement. Try it without the quotes to see what happens. This should produce something like this:

This is your first error message.

Assignment 5.4 Read the chapter from our text titled, "Chapter 1 - The way of the program". While you read the chapter open the Python environment through the Linux terminal on your Raspberry Pi as before. Follow along with the examples from the text as you go. This will help you learn Python more quickly. When you are done reading the chapter complete exercises 1 and 2 at the end of the chapter on your own to make sure you understand what you read. If not, see the Note above.

5.5 Functions

As our text states,

> In the context of programming, a function is a named sequence of statements that performs a computation. When you define a function, you specify the name and the sequence of statements. Later, you can "call" the function by name.

Let's take a look at this sequentially:

First, define a function. Second, call the function.

What does this look like?

Open the Python environment window and define your function like this:

Then we call the function like this:

So, what happened here? We defined the function `say_hello()` . Then, we called the function and passed a parameter into the function. Can you tell what the parameter is?

Why do we need to know how to use functions?

The first reason is that the standard Python 3 library is filled with functions. When you type the command `print ("Hello, World!")` you are calling the built in Python `print()` function. As you read through the chapter on functions you will see that there are many other examples of useful functions packed into Python.

The second reason is to make your programs easier to use. For example, let's say you make a robot that needs to follow this pattern as it drives alone a course. There are many reasons why you might need to do something like this. Maybe the problem to solve is navigating around obstacles. Maybe the robot needs to map a facility that humans cannot get into because of hazardous conditions like radiation. Maybe your robotics teacher just wants you to demonstrate you can use functions. At any rate, functions will make your job as a programmer eaiser. Here is the path the robot will take:

How many times does the robot need to make a 90 degree left turn? (nine) Here is an example of how we could execute that turn using Python and our Raspberry Pi (Don't worry about the syntax now, we will get there.)

As you will learn, there are other commands that need to be used to set up this program but these four lines are what actually make the Raspberry Pi control the left and right wheels of the robot so that the right wheel moves forward for two seconds while the left wheel stays fixed. This will make the robot turn left. If the two seconds is the correct amount of time, it will make a 90 degree turn.

Think about writing the program to just make the robot drive around. You would need to replicate this code nine times. You probably would not retype it nine times. Instead you would copy and paste it. This is known as brute force programming. There are many reasons not to do this. The biggest reason has to do with the third line in our left turn program. What if two seconds is not exactly right? If that is a slight overturn then the robot will go off course with every turn which means we need to go back in and change that value on every `sleep()` function. What if the battery on the robot is at a different charge level from one day to the next? Again, we will need to change nine lines of code. Remember, we humans are not very good at repetitive tasks. That is what we use computers for - to do things over and over. Why not let the computer do the repetitive stuff?

Instead, let's make a function called `left_turn()` like this:

And call it nine times throughout the program with the command `left_turn(2)` . If we found that it actually takes three seconds instead of two to execute a 90 degree turn then we would just need to change the parameter we pass to the function, represented by the variable `sleep_time`.

In that case, the call would look like this: `sleep_time(3)`. We could even make things simpler yet by defining a variable at the beginning of the program to represent the number of seconds we want to sleep during the turn - something like `turn_sleep_time = 2` . Then, we would call the function like this:

```
left_turn(turn_sleep_time)
```

Now, every time we change the variable `turn_sleep_time` we are changing the parameter passed to the function everywhere in the program. We have gone from changing nine different lines of code to changing just one. This gets really impressive when we think about all the right turns we have to make as well. See how functions are an integral part of robotics? Every modern programming language uses functions. Once you learn it for Python you can use it in other languages as well with only minor changes to syntax.

If we write programs that need to repeat a process nine times, functions save us a lot of work and potential errors. If we need our robot to do something nine hundred times or nine million times we have to use them. Now let's spend some time learning more about the functions built in to Python using our text:

Assignment 5.5 Read the chapter from our text titled, "Chapter 3 - Functions". While you read the chapter open the Python environment through the Linux terminal on your Raspberry Pi as before. Follow along with the examples from the text as you go. This will help you learn Python more quickly. When you are done reading the chapter complete exercises 1, 2 and 3 at the end of the chapter on your own to make sure you understand what you read. If not, see the Note above.

5.6 Conditionals

Besides completing repetitive tasks for us humans, robots need to be able to make decisions based on the condition of their environment. Later, we will learn how to use sensors to understand what is happening in a robot's environment. First, we will learn how conditional statements work in Python. If you have not already done so, it might be helpful to get a little background on computer logic. The inventor of the logical system we use in robotics was George Boole. There is a video in the Supplemental Videos section of this book about his contributions to the field.

For now, we will concern ourselves with some basic logical or Boolean operators. These are different than mathematical operators. When we write the following statement using mathematical operators:

```
x = 6
```

we are making a statement that the variable `x` is equal to the integer 6. By contrast, when we use the same operator (=) in logical math we are asking "Is one equal to the other?" We script that questions like this:

```
x == 6
```

Notice that we use 2 = signs instead of one. This is not a typo. In fact, it is universal across many programming languages. Instead of declaring x is equal to 6, it is asking Is x equal to 6? What are the possible answers to this question? For the answer, read section 5.2 of our text.

Notice that there are many other logical (aka relational) operators. We can use these operators which return a value of true or false to help our robots make decisions. To do this we use functions like the `if()` statement. Let's try it out using another Python function `len()`. We will create a function that determines if a word is longer than seven letters. If so, a message will be displayed saying so. If not, a different message will be displayed. Before you read how to do this, see if you can figure it out. Follow these steps using the textbook as a guide:

Define your function and pass a parameter through that is the word you are going to test.

Write a conditional statement using the `len()` Python function to determine the length of the word.

Use the `print()` function to return a statement that says if it is or is not a long word (greater than seven letters long).

See if you can include the word in the message returned by the function

Here is one way to do it:

Notice that we pass the word to the function by using the parameter `word` which then stores the string representing our word as a variable. The if statement checks the length of the word using the `len()` function which returns an integer

equal to the number of alphanumeric characters in a string (our word). If that value is greater than or equal to nine, one message is printed. Since the only other possibility for the length of this word (if it is not greater than or equal to nine) is that it is less than nine letters, we can use the `else` function to print a different message. Try it out now by typing this function into the terminal under the Python environment and calling the function using different words. You will probably get some "Traceback" errors the first time you do but stick with it. When you are done compare your result to the results below:

Think of the possibilities now. What could your robot do with this new power! If we go back to our example in the previous section in which our robot had to navigate around obstacles by turning 90 degrees at a time we can see some interesting solutions. If we could use a compass sensor to determine the angle of our robot in relation to magnetic north, we could use conditional statments to turn the robot. Instead of trying to figure out exactly how long it takes for the robot to turn 90 degrees, we could write a script that waited until the angle in relation to magnetic north changed by 90 degrees. We could even do this with one function. It might look something like this:

This is a lot to take in for a new programmer so let's briefly go over this function. When it is called we put in a value for direction. The possible correct values are "left" and "right". If the direction is "right" the left wheel moves until the angle of the robot is equal to the desired angle. If "right" is passed to the function the opposite happens. Just in case the human mispells the word or puts something else in a third print statement is used to tell them that their input is not valid. We will get to this process later in the course.

Assignment 5.6 Read the chapter from our text titled, "Chapter 5 - Conditionals". While you read the chapter open the Python environment through the Linux terminal on your Raspberry Pi as before. Follow along with the examples from the text as you go. This will help you learn Python more quickly. When you are done reading the chapter complete exercises 1, 2 and 3 at the end of the chapter on your own to make sure you understand what you read. If not, see the Note above.

5.7 Iteration

There are many reasons to use iteration in programming. The most straightforward reason is to do a spedified action a certain number of times. Imagine you are designing a robotic NASCAR driver. This would be the perfect place for iteration. In a NASCAR race the driver makes the same left turns hundreds of times. Considering one of the advantages of robots is that they can perform tasks that humans find boring or too repetitive, our robot NASCAR driver would thrive in this environment making the same left turns over and over for hours.

We will approach iteration by learning two programming concepts: `for` loops and `while` statements. Each is useful in various situations. Sometimes they are interchangable depending on the problem being solved. One uses Boolean logic and the other does not. Let's start with the `for` loop.

5.7.1 `for` Loops

Unitl now, everything we have done, we have done once. Most of the time though, our robots will need to do things repeatedly until a desired outcome is reached. An example could be that our robots collect data from a website over the course of several hours. One of my home automation projects was to build a solar powered garden watering system for my wife's garden. This is very much a robotics application as we wanted to make sure water is delivered consistently every day even when we were gone on vacation. One of the challenges though was that sometimes it rains in the summer, ususally at night. If it rains enough the night before there is no need to water the next morning. As humans we would just think, "How much did it rain yesterday (if it did at all)?" and make our decision to water that morning. A robot needs clear rules though to follow. In this case, after a lot of trial and error, I decided that it was best to check the weather conditions between 4:00 pm and the time the watering system came on in the morning. That way, if enough rain had fallen there was no need to water the garden. I decided to check the weather forcast using openweathermap.org. Every 15 minutes between 4:00 pm and 8:00 am the Raspberry Pi would "scrape" the current weather data off the website, "parse" out the current weather condition and record it in a csv file. To do this I used something that looked like this:

Don't worry too much about the details of this function called `collect_weather()`. Instead focus on the line that begins with the word `for`. This is one way to use a `for` loop to iterate a process (checking the weather). I know that I want to check the weather over the course of 15 hours every 15 minutes (four times an hour). This loop will start at 4:00 pm and run 60 times. During the loop, the Pi checks the weather and records the data in a file then sleeps for 15 minutes (900 seconds). This is a very complicated script so let's try something simpler using our Raspberry Pi and the Python environment.

Open a ternminal and start the Python environment as before. Type the following lines:

This `for` loop will count to five, updating the value of `x` each time and print `x`. Try it out by hitting enter to get this result:

Notice that, while the loop ran five times, the value of `x` was not five at the end. You can see why. In computer science we generally start counting at zero instead of one. As a result the `range()` function used in a `for` loop ends its final iteration at one less than the value passed to the function, in this case four instead of five. Remember this for future scripts.

There are other ways to use `for` like iterating over lists. We will get to those later in our course. For now, explore this method by completing the next assignment.

Assignment 5.7.1 Read sections 1-3 of chapter 4 from our text. While you read the chapter open the Python environment through the Linux terminal on your Raspberry Pi as before. Follow along with the examples from the text as you go. This will help you learn Python more quickly. When you are done reading the chapter complete exercises in section 4.3 on your own to make sure you understand what you read. If not, see the Note above.

5.7.2 `while` Loops

Our last introductory programming concept is a different kind of loop. Sometimes, the programmer does not know how many iterations will be necessary to solve the problem. For example, from the previous description of a Raspberry Pi based watering system for a garden, I did not know when I would stop using the system. At the end of the season after the first freeze we would have no more need to water. At that point, we would just shut it down and take the electronics in for the winter. Until then, I needed it to keep going. As a result, I used one of the simplest loops which is written like this:

```
while True:
```

`while` loops run as long as the condition being tested is true. In this case, I am saying do everything in the loop as long as true is true. This will be true forever (or until I unplug the Raspberry Pi for the season). There are many other ways to use `while` loops but the basic concept is the same - keep doing these things until the condition is no longer true. For example, let's say you are building that robotic NASCAR system mentioned earlier. If you wanted to keep the car going until the end of the race you could say something like this:

This extremely oversimplified script basically says, we start the race at zero laps. `if` we do something that means we have completed a lap, add one to the total laps. So, after the finish line is crossed the first time, laps gets increased by one. `while` laps is less than or equal to 500 (the end) keep driving.

To understand this more, complete the next assignment.

Assignment 5.7.2 Read the chapter from our text titled, "Chapter 7 - Iteration". While you read the chapter open the Python environment through the Linux terminal on your Raspberry Pi as before. Follow along with the examples from the text as you go. This will help you learn Python more quickly. When you are done reading the chapter complete exercises 1 and 2 at the end of the chapter on your own to make sure you understand what you read. If not, see the Note above.

Chapter 6

Robotic Types

A fundamental qustion in mechatronics is, "what is a robot?" Over the course of this year, you should be able to come up with some answers for this question. This short section is aimed at helping us understand that before moving on to our final project. Now that you have learned basics of electronics, mechanics and programming, you may begin to explore the larger field of robotics applications. For this assignment we will begin with background from some experts to supplement what you have already learned.

To begin, let's take a look at some of the most recent robotics development using a very thorough BBC documentary:

Make a list of your own criteria that make something a robot. This list may include anything you like. Give a justification for each criteria.

Read this page from the IEEE which attempts to help us decide what a robot is and is not. How does this change your list? Be sure to listen to the speakers on the page.

Assignment 6.0 Choose one of the robot types from this page. Conduct your own reaserch and create a 5-10 minute presentation demonstrating the characteristics and applications of your chosen robot type. Provide video descriptions of these real-world robots as well as the educational pathways needed to work with these robot types.

Chapter 7

Capstone Project

The stone placed at the very top of an archway which holds the rest of the arch in place is known as the capstone. Just like mason placing that last stone, you are ready to demonstrate your learning. For the last four months of this course you will complete a capstone project.

You, and a partner if you choose, will complete a complex robotics project using everything you have learned this year.

Overview:

Each group/person must design, build, test, and demonstrate a device controlled by one or more microcontrollers (e.g. Arduino, Raspberry Pi, ESP 8266, etc.). The device should have functioning elements in all five categories listed below. The device will be rated (graded) based on the level of functionality achieved in each category. There will also be grading adjustments for qualitative attributes.

You are allowed to use circuits and code you find on the Internet or in other books, but make sure you have a full understanding of what the code or circuit does. You should also "add value" to what you use by incorporating it into your system creatively and effectively. Also, make sure you cite in your report, your poster and in your final code sketches any sources of any code or circuits you use.

The materials used for this project must meet the following requirements:

Any materials from the Arduino Super Starter kit for class may be used including sensors, motors, wires, lights, displays and resistive devices

Any approved components available in the lab (i.e. sensors on loan, stepper motors, structural parts, etc.)

Up to 200g of ABS filament (for entire project, including prototypes).

Up to $75 in pre-approved student purchased parts (per project, not per student). It is up to the student to order these components although I will help

you find them, if needed. Be careful here, qualitative deductions will be made for expensive and/or unnecessary component selection (i.e. using a Raspberry Pi where an Arduino UNO will work, etc.)

NOTE - the choice of your project concept could have a large impact on the grade you receive; so please evaluate your alternative concepts carefully, based on the grading criteria below.

Each of you will receive a detailed set of instructions for this project.

Chapter 8

Going Further with Robotics

8.1 Robotics in Higher Education

You are taking this course because you are interested in learning about robots, obviously. You may want to pursue this subject after you leave high school as well. What does it mean to be a robotic engineer or roboticist? This is more than having an enthusiasm for the field. It means pursuing an educational path that prepares you for a profession in robotics. It also looks different depending on what you want to do with robotics. Robotics engineering is a standalone field within mechatronics. However, many fields benefit from this area of study even if it not the primare goal. One of our recent alumns, Katie Schutt, summs up her path through robotics at Windsor High School and how it helped her get to Aeronautical Engineering:

> WHS robotics was the high school program that probably best prepared me for life and education in college, especially as an incoming engineering student. I learned technical skills around designing and testing under the pressure of requirements and a deadline, and the necessary programming logic and proficiency. Equally important, I developed skills in public speaking, documenting the proposals and then progress of my projects, and the ability to constructively work with a team. All of these skills were heavily emphasized and required to succeed in my first semester at CU, and robotics provided the foundation for those skills to be built upon.

Katie is working to become an astronaut with the goal of traveling to other planets in our solar system. The technical skills she is building will give her opportunities like this. It began in her robotics classroom at WHS.

You may not know where you want to take robotics yet. There are hundred of branches of robotics. All of them have some common threads. In high school some of these skills may be learned to prepare students for university level robotics:

Math

Algebra

Geometry

Trigonometry

Calculus

Physics

Dynamics

Kinematics

Computer Science

Programming languages

Computation logic

Cloud Computing

Engineering

Language Arts

To get on this path be sure to talk to your counselor about your goals. Feel free to ask me about this as well. We have many former students who have pursued this field and are attending colleges all over the United States. An excellent way to learn more is to attend an engineering day at Colorado State University or the University of Colorado in Boulder. Most high school students find that a visit to campus makes the potential college experience more attainable and realistic.

8.2 Supplemental Videos

The Genius of George Boole

Shakey:Experiments in Robotic Planning and Learning (1972)

BBC Documentary - Hyper Evolution : Rise Of The Robots (Part 1)

BBC Documentary - Hyper Evolution : Rise Of The Robots (Part 2)

BBC Documentary - Hyper Evolution : Rise Of The Robots (Part 3)

Robots that fly … and cooperate | Vijay Kumar

Chapter 9

Course Materials and Instructions

9.1 Course Lecture Notes

9.1.1 Introduction

9.1.2 Electronics

9.1.3 Programming

9.1.4 Robotics Concepts

9.2 Using GitHub

For this couse we will store all Python and Bash scripts on GitHub. This will alllow us to utilize the version control system Git and work from an online repository instead of storing all code on a single device. In doing so, you will be able to collaborate with others more easily and use multible devices on the same project such as a PC at school, a Mac at home and a Raspberry Pi that travels with you.

Sign up for a GitHub account Go to GitHub and choose “Sign up”. Create and verify your account. Be sure to record your credentials somewhere for later.

To understand what GitHub is and how we can use it with any programming language complete the Hello World Tutorial. You should also watch this introductory video as well:

9.3 Think Python Textbook

Our course programming text is called Think Python by Allen B. Downey. It will most likely be easiest to use the HTML version but there is also a PDF download at the site as well.

9.4 All About Circuits Textbook

Our course electronics text is Volume I of All About Circuits. This online text will focus on Direct Current (DC) electricity.

9.5 OnShape Account Setup

OnShape offers a powerful online CAD solution. Students in secondary education can receive a free student account which includes all of the features of the paid account and the learning center. This makes it an outstanding CAD alternative.

Go to the OnShape website to sign up for a free account. Be sure to use your school email address in the process.

9.6 Controlling the Raspberry Pi Remotely

Using a remote shell, one can control a Raspberry Pi with another computer. You can do this through the linux command line or with a GUI using a service like Tight VNC. Learning how to do this will allow our robots to become less tethered to a specific location. This concept, more broadly is known as the Internet of Things (IoT). Devices with internet connections can interact with one another over that connection. Many of you probably already have some IoT devices in your homes like thermostats that interact with each other to maintain the most energy-efficient comfort in your house to security cameras that may be accessed on a mobile device from anywhere. In short, our robots will not be fully robots until we learn to do this. As a result, the Capstone project for this course in semester two requires this.

In all cases one thing is assumed. The Raspberry Pi and the device using the Secure Shell are connected to the same router. In our lab, we will use the MechLab wifi network. At home, students will need to make sure to connect the pi to their home router which the computer is using.

This process will work for a computer running Windows, MacOS or Linux. It will not work for a Chromebook.

9.6.1 Setting up SSH for Windows

Complete the following steps to create a secure shell connection with a computer running Windows:

Be sure the Raspberry Pi is turned on and connected to router.

Be sure the computer is connected to the same network.

Open a terminal window on the Pi and type `ip address show`.

You will see something like this:

Make a note of this address, this is the address of the Pi on the network. The computer will also have an ip address on the network. To see that address open Windows PowerShell and type `ipconfig`. The "IPv4 Address" is the address of the computer. If both are on the same network, the first three numbers should be the same. If not, they are not connected to the same network and a SSH connection cannot be made.

Download and install Putty (Most will choose the 64 bit Windows Installer file).

Open Putty.

In the "Host Name (IP address)" box, type in the address of the Raspberry Pi.

Click "Yes" at the warning.

Enter the username and password for the device.

You are now connected to the Pi via SSH to the command line.

If you would like to set up a graphical user interface (GUI) for use with SSH you can try using TightVNC. This HowToGeek Guide shows you how to do it.

9.6.2 Setting up SSH for Mac or Linux

In case you need to use Mac or Linux for your host computer, the process is very similar for each. Follow this guide to set it up.