IoT in Project-Based Learning

Dr. Hugh Jack P. Eng., Western Carolina University

Dr. Jack is the Cass Ballenger Distinguished Professor of Engineering in the School of Engineering + Technology within Western Carolina University. His interests include robotics, automation, and product design.

Mr. Adam Harris, Western Carolina University

IoT in Project-Based Learning

Abstract

At Western Carolina University, a four-year Project-Based Learning (PBL) sequence is implemented across the undergraduate residential programs, culminating in a senior capstone project for industry partners. This interdisciplinary approach enables students to develop professional skills in project management, ethics, design, teamwork, and more through hands-on projects. With diverse team compositions, students focus on various technical aspects, such as Computer-Aided Design (CAD), 3D printing, assembly, testing, electrical, and software components.

This paper highlights the incorporation of an Internet of Things (IoT) platform into the second-year PBL course. Students were introduced to a low-cost ($6) microcontroller with wireless internet access and provided with prototype Python programs. These programs enabled students to create their own wireless access points and simple web servers, facilitating interaction with the microcontroller and their projects using phones or laptops. By connecting the microcontroller to sensors and actuators, students were offered a wide range of design possibilities. The paper provides sufficient detail for educators to adopt a similar approach in their courses, promoting interdisciplinary learning and hands-on experience with IoT technologies.

Introduction

Western Carolina University has programs in Engineering and Engineering Technology in the same school. The programs include;

- Electrical and Computer Engineering Technology
- Electrical Engineering
- Engineering with concentrations in Mechanical, Manufacturing, and Electrical Power
- Engineering Technology - Applied Systems
- Engineering Technology - Technical Operations (off-campus distance)
- Master of Science in Engineering Technology

The school has a focus on practice and strives to prepare students for industry. To achieve this goal the school has adopted a Project Based Learning (PBL) core that has a common core of five project courses, with at least one in each year of the program. The first two courses, ENGR 199 and 200, focus on the basics of project work and act to level students' abilities. ENGR 350 has a focus on entrepreneurial product design. ENGR 400 and 450 are the capstone project courses.

The capstone projects are done for industry and typically include product design, test equipment, production equipment, or process research.

ENGR 199 - Introduction to Engineering Practices and Principles I, Credits: 3
ENGR 200 - Engineering Practices and Principles II, Credits: 3
ENGR 350 - Engineering Practices and Principles III, Credits: 3
ENGR 400 - Engineering Capstone I, Credits: 3
ENGR 450 - Engineering Capstone II, Credits: 3

In addition to supporting the mission of the school, it is worth noting that these courses also provide a common base for ABET assessment of the EAC and ETAC programs. The topics covered in the courses are introduced and reinforced in subsequent courses. Topics include;

- Project Management - Work Breakdown, Gantt charts, budgets, risk management
- Professional Tools - Microsoft Project, Teams, Excel
- Professionalism - Ethics, Teamwork, Communication
- Design - Needs identification, ideation, concept generation, iteration
- Structure - Requirements, specifications, drawings, proposals
- Multidisciplinary - Projects mix elements from electrical, mechanical, computer, and manufacturing domains
- Context - Global issues, standards, contemporary issues, customers

The project described in this paper was conducted in ENGR 200 in the Fall of 2022. It is normal for the students in the class to have a variety of expectations, motivations, and backgrounds. Some had previous exposure to programming including platforms like the Arduinos. Others had no experience with programming or electrical work. Some had previous exposure to Computer Aided Design (CAD) systems, but this too was inconsistent. The class also included a mix of students that had taken ENGR 199 at [[[XXX]]] and other students that had transferred equivalent courses from other universities and community colleges. With a multidisciplinary class from all programs, some students were inclined to identify as mechanical/manufacturing versus electrical/computer. As a result, many complex class exercises were run in a team format and students were encouraged to select teams that blend design skills. A secondary benefit of this approach was to encourage socialization and build teams of students after the pandemic.

The discipline-specific technical content of the courses is secondary to the PBL topics. Therefore, it is permissible to allow students to have areas of strength and weakness. Topics from each of the disciplines are introduced to provide a baseline for project work. For example, in the Fall of 2022, the class was taught Autodesk Fusion 360 [12] for CAD work and 3D printing. This was of direct benefit to students in the Mechanical/Manufacturing programs. The class taught basic programming skills using Python. This was of benefit to all students. And, the class was taught to interface the Raspberry Pi Pico. This directly benefited students in the

Electrical/Computer programs. This paper focuses on the use of the Pico for networked applications.

The Internet of Things (IoT) technologies have been emerging for some time. Although the term IoT was coined in 1999 by Kevin Ashton [14], the development had begun before. An early, possibly the first, internet device was a coke vending machine at Carnegie Melon University [13]. A more recent example is an internet-connected coffee maker with a web camera [9] and an internet standard for controlling such devices [10]. People now take these capabilities for granted. Consider the common home automation system where voice control can be used to control lighting [11], or users can receive a message on their phone when someone rings a doorbell [12].

The common components in an IoT platform are listed. As a formal area of study IoT includes communication protocols, security, power consumption, control algorithms, applications, and more.

1. A process interfaced with sensors and possible actuators.
2. A microcontroller connected to a network.
3. A 'cloud platform' to remotely receive and process the data.
4. Software at the microcontroller and cloud levels.
5. Additional elements might include Web or App interfaces for users (and much more.)

The Hardware Platform

There are many excellent choices for embedded systems in lower-level engineering courses. Over the years the author has made good use of Arduinos [8] and custom hardware [6][7]. Over time the cost for these platforms has dropped consistently to the point now when they are basically disposable components.

In this recent application, the Raspberry Pi (RP) Pico W [1] was used. Some of the features of this microcontroller are:

- Dual Core ARM processor at 133 MHz
- A Wifi module that can act as a client or server
- 2 MB of flash memory
- 264 KB of Ram
- 26 multiuse IO pins at 3.3V

In addition, a $30 'Pico Explorer' expansion board was used to add a color display, buttons, a buzzer, and two motor drivers [15]. While this particular board was not required, it did simplify usability and prototyping. An Explorer and RP Pico W were assigned to groups of 2-3 students

for the semesters. Figure 1 shows a Raspberry Pi Pico W module at the bottom of the figure. In the upper half of the image a Pico is mounted in an explorer board. The explorer board has an LCD screen on the right, 4 push buttons, H-bridge motor drivers, a piezo element, and a small breadboard.

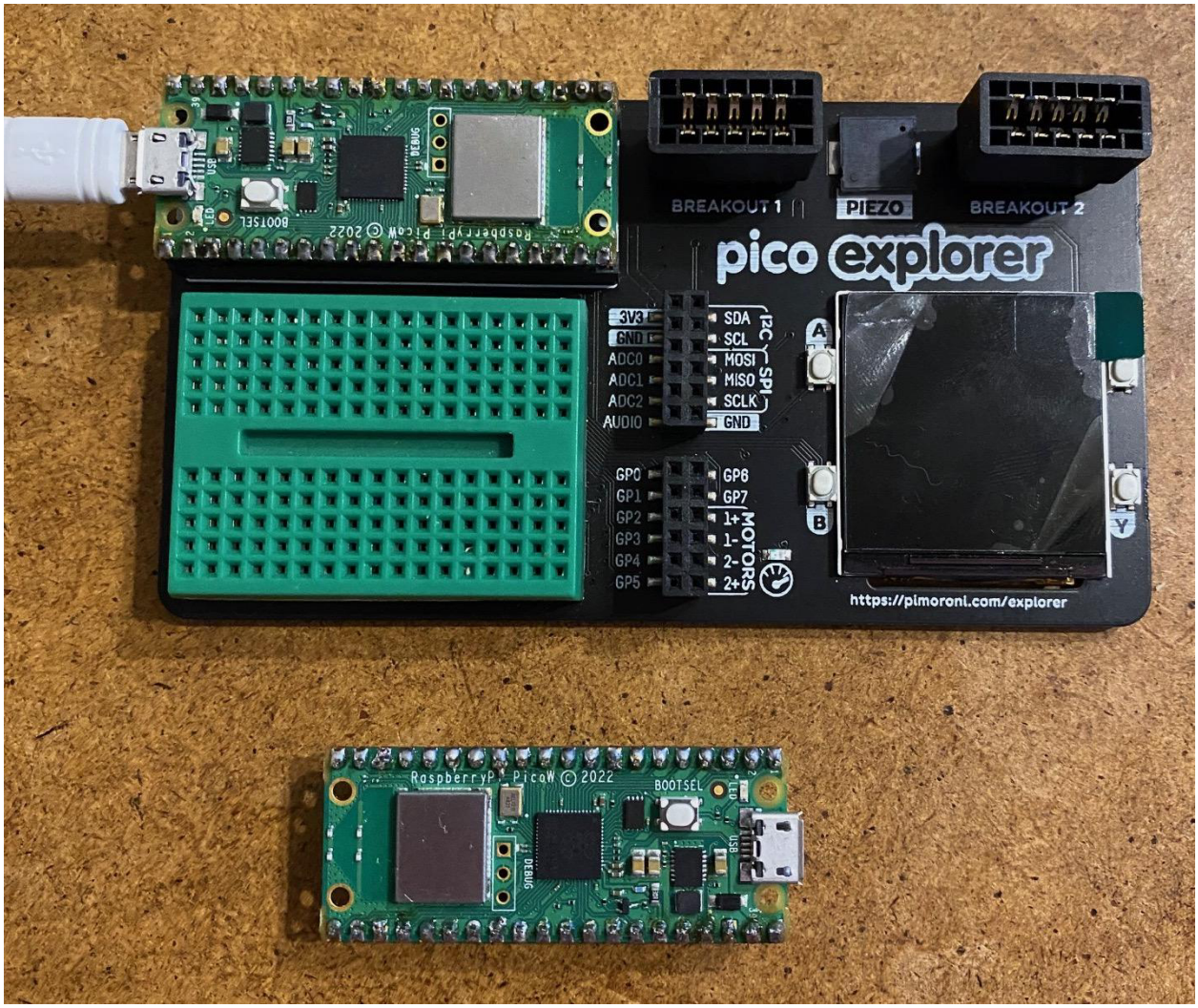


Figure 1 - A Raspberry Pi Pico W by Itself and Mounted in an Explorer Board

It is important to note that these Pico boards are not the same as the regular Raspberry Pi platforms. The regular RP boards have more memory, storage, and other capabilities. These boards are meant to be microcontroller replacements and are not capable of running a modern operating system like Linux. Instead, this project made use of Micropython [2] as the programming platform, although there are other great options including Circuitpython [3] and C/Arduino [4]. In addition, there are some good options for the programming Integrated Development Environment (IDE.) In this case, Thonny [5] was used for programming. Programs could be stored on the Pico and/or computer and be downloaded as needed. Many students kept

their programs on the Pico only and edited the code using their laptops. When the Pico was run in a standalone mode the programs could be loaded and run from the 2MB of flash memory.

The Pedagogy

The use of Python on the Pico was introduced over a few weeks. It was understood that all students needed to have some grasp of Python programming, the experience included individual homework with simple electrical connections to the Pico, e.g. Light Emitting Diodes (LEDs). More advanced use of the Pico allowed select members of the teams to specialize in the electrical and software elements. The assignments for the semester are listed for reference with some description. Note that weeks that were lecture oriented overlapped with project work so students were always working on projects.

Week 1 - CAD and 3D printing (Individual Work)
Week 2 - Python and a number guess program (Individual Work)
Week 3 - Micropython and an LED flashing program (Individual Work)
Week 4 - Micropython and IO with a mechanical aspect (Group Work)
Week 5 - Servos with the Pico (Group Work)
Week 6 - A web server with the Pico (Group Work)
Week 7 - Project Management Introduction (Individual Work)
Week 8 - Project Management Continued (Individual Work)
Week 9 - Gantt Charts and Microsoft Project (Individual Work)
Week 10 - Project Needs and Requirements (Individual Work)
Week 11 - Design Concepts and Selection (Individual Work)
Week 12 - Risk and Critical Thinking (Individual Work)
Weeks 13-15 Project Work (Group Work)

Students were introduced to the Python language with an online tutorial guide supported by a simulator [16]. Given that the objective was not to teach a programming language, but to have them deploy design functionality, they were allowed to access other resources and even use source code from websites. They were given assignments for a number guess game to provide the concepts of loops and if statements. A following assignment had them write a program to flash the S-O-S pattern in Morse code on an LED. Surprisingly, many of the students provided unique and functional solutions, even though they were allowed to collaborate with their classmates.

The first complex program developed with the students is shown in Figure 2. They were encouraged to enter the code in their own laptop and try it on the board themselve, even though they were sharing the boards with other students. In the program they learn the concept of importing modules/libraries. The machine module was used to configure and address hardware

for an LED, pushbutton, and speaker. Note: the outputs were connected by the students in groups. The concept of defining functions was used as a persistent technique to encourage students to modularize their programs. By using functions, the main part of the program is a while loop towards the bottom of the program where the button is checked to choose between flashing and LED or creating a tone. The physical interaction with the program is very powerful and gave the students more appreciation of the abilities of programming. This example was followed with additional examples like servo motors, providing that bridge between the electrical/computer world to mechanical equipment.

```
# a flashy light program

import machine
import time


LED = machine.Pin( 6, machine.Pin.OUT )
button = machine.Pin( 15, machine.Pin.IN, machine.Pin.PULL_UP )
speaker = machine.Pin( 7, machine.Pin.OUT )


def flash( frequency ):
  LED.value(1)
  time.sleep( 0.5 / frequency )
  LED.value(0)
  time.sleep( 0.5 / frequency)
```

```
def tone( frequency ):
    speaker.value(1)
    time.sleep( 0.5 / frequency )
    speaker.value(0)
    time.sleep( 0.5 / frequency )

start_f = 100

while 1 == 1:
  if button.value() == 1:
    flash( 50 )
  else:
    start_f = start_f + 1
    if start_f > 15000:
      start_f = 100
    tone( start_f )
```

Figure 2 - A Basic Python Program for the Pico

One the students were acquainted with the hardware and software it was time to move onto internet connectivity. The first major leap was to show students that their program was able to create a wireless access point. The function in Figure 3 was provided to them with a 10 minute step-by-step explanation, but they were allowed to treat it as a black box. However, students in electrical and computer programs were encouraged to put more effort into understanding before they formally covered the content in other courses. Technically, the Pico can be used as a normal network client, or as an access point. As an access point it appears as another wireless network. Students can see the network name and connect from their laptops and phones using the password. Being able to create their own networks was very empowering to many students. When the students ran the example below it would create the access point and allow them to connect from their devices, but the python program would quit because setup_webserver and server_loop were not defined yet. By the time they were done they were aware of IP addresses of the web server, 127.0.0.4.

```
import network
import socket
# this function is called to get the access point started.
def setup_access_point(ssid, password):
    print('setting up the device as an access point')
```

```
    ap = network.WLAN( network.AP_IF ) # Want a wireless network access point
    ap.config( essid=ssid, password=password ) # the unique network stuff
    ap.active(True)

    while ap.active == False:
        pass # this does nothing, but it adds something inside the while loop
        # oddly, using anything else inside the loop leads to problems
    print("Yay: Access point active")

    config = ap.ifconfig() # get some data
    server_address = str(config[0])
    # people should be able to connect to the ssid now with the password
    print("Connect using IP address " + server_address ) # print the IP address
    # print(ap) # we could dump more data just for fun

# main program loop
ssid = 'engr200'
password = 'secret1234'
setup_access_point( ssid, password )
s = setup_webserver()
while True:
        server_loop(s)
```

Figure 3 - Setting Up an Access Point on a Pico

After the access point was operating students were ready to create a simple web server. This process required three functions. The first sets up a standard web/http port (80) and creates a socket to listen for incoming connection requests. This provides an opportunity to describe the range of ports and computers and their functions. The students enter this before the main program loop.

```
def setup_webserver():
    # this sets up port 80 (http) to listen for incoming requests
    s = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
    s.setsockopt( socket.SOL_SOCKET, socket.SO_REUSEADDR, 1 )
            # this fixes an error EADDRINUSE
    s.bind( ('', 80) ) # this is the port (socket) for http/web
    s.listen( 5 )

    print('socket set up and we are now ready for web connections')
    return s
```

Figure 4 - A Python Function to Setup a Webserver Socket and Start Listening

This leads to a discussion of what happens when you connect to a web server. The function in Figure 5 accepts the server socket (listening on port 80.) The print statements before and after helping the students understand what is happening in the terminal in Thonny. When the program is run, they see a message saying that they can connect to an IP address and it will proceed to the

line ‘Waiting for a new connection’ and stop. They are then told to connect to their access point with their phone. Then they are told to open a web browser and go to the address http://127.0.0.4. As soon as they enter the text the ‘Got a connection…’ message appears with the IP address of their phone. Even better they see the raw ‘GET \’ command from the web browser with all of the other related information like browser type. This provides a great transition discussing the response from a web server.

```
def server_loop(s):
    print('Waiting for a new connection')
    conn, addr = s.accept()
    print('Got a connection from %s' % str(addr))
    request = conn.recv(2048) # this grabs up to 1024 characters

    print('      ')
    print('The following came from the remote web browser')
    request_string = request.decode() # convert the characters to
string
    print(request_string)
    response_string = request_decode(request_string)
         # if request is 'GET index.html' we should get the file back

    print('   ')
    print( 'we are returning ' + response_string)
    conn.send('HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n')
         # Web browsers like to see this
    conn.send(response_string)
    conn.close()
    print('    ')
```

Figure 5 - A Main Webserver Loop

The function to decode a request is shown in Figure 6. In this case any ‘GET’ function will get the ‘I Don’t’ response, regardless of which files or directory they request. If they do something odd, they will get the second one. Students can change the HTML and you can show them some simple tricks like making lists, headings, italics, etc. But, when the students get the response you can look at the raw HTML. Most browsers have a ‘developer/view source’ option. At this point the students understand the process as a two-way street.

It is possible to cover the process from figures 3 to 6 in a one-hour class if you rush, or at a relaxed pace in a two-hour course. Students leave with a better understanding of web connections. The next stage is to introduce them to the concept of a Rest API. In the interest of brevity, a more complete program is provided in Appendix C, followed by HTML code in following Appendices. This code allows more interaction with the Pico to change parameters and more.

```
def request_decode(request_string):
    request_list = request_string.split(' ')
    print('starts with _' + request_list[0] + '_' )
    if request_list[0] == 'GET':
        response_string = “””
            <HTML>
            <BODY>
                  I don’t do filenames, this is all you get
            </BODY>
            </HTML>
        “””
    else:
        print('Ouch, I do not know what to do with this')
        response_string = “””
            <HTML>
                  Oof, I don’t know what to do with that
            </HTML>

        “””
    return response_string
```

Figure 6 - Decoding a GET Command

The Projects

Given that the purpose of the course was PBL, and to serve multiple disciplines, the projects were interwoven with the lecture work and homework. Two major group design and build projects were done to use the IoT knowledge along with mechanical design and build. Full requirements for the projects are provided in Appendix A and B. But briefly, the first project lasted 4 weeks and was Halloween themed. Students were encouraged to develop an interactive device to ‘scare’ trick-or-treaters. This was the first change to integrate multiple disciplinary skills, including the Pico boards.

Weeks 6-9 - A Truly Scary Halloween Project - See Appendix A for details
Weeks 10-15 - Tunnel of Love - See Appendix B for details

The last project lasted for 6 weeks and comprised of a two-part design for each team of four students. Students were task with making a device to launch balls at a moving target, and the team also made their own moving targets. They then competed against all the other teams in the class. All the devices were required to be remotely controlled with no wires or other physical connections. Almost all the devices were based of the Raspberry Pi Pico boards with modified Python programs. Students used their laptops or phones to connect to their devices and steer the

cars or aim and fire the balls strategically. Notably, all six of the ball throwers worked correctly. And, only one of the 6 cars did not work.

Figure 7 - Ball Throwers

Figure 8 - Moving Targets

Conclusion

The introduction of Internet of Things methods into a second year PBL course went remarkably well. Using the black box approach as the minimum expectation, but looking at the details, allow some students to explore intellectually, while others did not explore but gained an appreciation of other disciplines. The author plan to continue using this approach but will refine the code to a cookbook of smaller functions that students can select and use as needed. More advanced students will be able to modify functions or write their own.

References

[1] “Raspberry Pi Pico”, https://www.raspberrypi.com/products/raspberry-pi-pico/, accessed February 13, 2023
[2] “Micropython”, https://micropython.org/, accessed February 13, 2023
[3] “Circuitpython”, https://circuitpython.org/, accessed February 13, 2023
[4] “Arduino-pico”, https://github.com/earlephilhower/arduino-pico, accessed February 13, 2023
[5] “Thonny Python IDE for Beginners”, https://thonny.org/, accessed February 13, 2023
[6] Jack, H., & Barakat, N. (2006, June), A Student Owned Microcontroller Board Paper presented at 2006 Annual Conference & Exposition, Chicago, Illinois. 10.18260/1-2--1244
[7] Jack, H. (2013, June), Using the Parallax Propeller for Mechatronics Education Paper presented at 2013 ASEE Annual Conference & Exposition, Atlanta, Georgia. 10.18260/1-2--22728
[8] Qi, H., & Jack, H. (2016, June), A Scalable Course Project to Accommodate Academic Variation Paper presented at 2016 ASEE Annual Conference & Exposition, New Orleans, Louisiana. 10.18260/p.27275
[9] “Trojan Room Coffee Machine”, https://www.cl.cam.ac.uk/coffee/coffee.html, accessed February 13, 2023
[10] Masinter, L. “Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)”, https://www.rfc-editor.org/rfc/rfc2324, accessed February 2023
[11] “Getting the most out of Matter with Alexa”, https://developer.amazon.com/en-US/alexa/matter, accessed February 13, 2023
[12] “ring”, https://ring.com/, accessed February 13, 2023
[13] “The “only” Coke Machine on the Internet”, https://www.cs.cmu.edu/~coke/history_long.txt, accessed February 13, 2023
[14] Ashton, K. (22 June 2009). "That 'Internet of Things' Thing". Accessed February 13, 2023
[15] “Pico Explorer Base”, https://shop.pimoroni.com/products/pico-explorer-base, accessed February 13, 2023
[16] “Python Tutorial”, https://www.w3schools.com/python/, accessed February 28, 2023

Appendix A - A Truly Scary Project Description

Halloween is coming! This year we are adding a new inventory item to scare our little visitors. Of course, we are engineers so we will build it ourselves. We set the following parameters to guide our design.

Requirements: (must be met to get 60%)

- Teams of 2-3 are to work on at least one device.
- The device should sit alone without cables or other connections. e.g., no USB cable for power.
- The device should be at least 6 inches x 6 inches x 6 inches. But it can be much larger if appropriate.
- The device should be partially or wholly controlled remotely by wifi or equivalent.
- (Obviously, there should be no damage or injury.)

Objectives: (will determine the scariness up to 40%)

- To elicit a fearful surprise reaction in a visitor.
- To be Halloween-themed.
- Include some exciting inputs and outputs.
- To make as much use of available resources and avoid spending money.

Deliverables: (must be submitted to assess objectives and requirements):

- Submit the documentation as a single doc or PDF file
- A cover page with a title, team member names, submission date, course name, etc.
- A one-page description of the design, major elements, and operation. Each team member must write their own.
- Programs with comments. This includes HTML files without comments.
- A set of pictures that show the major design elements.
- A table of components including quantity, description, source, and estimated cost.
- Images of any 3D printed parts.
- Other items as needed to illustrate the design.

Available Hardware:

- Raspberry Pi Pico and Explorer boards
- 3D Printing equipment
- Tools and small parts in the Makerspace
- Standard servos
- Other parts are available by request including,
    - Ping sensor (ultrasonic distance)
    - Continuous servos
    - Buttons and switches
    - LEDs
    - Sound sensor
    - Buzzer (fixed tone)

Suggestions:

- Avoid batteries, a USB charging pack is a great alternative.
- Old toys can be an excellent source of parts.
- The hardware section of Lowes can be a good trip for mechanical ideas.
- Consider sites like Thingiverse.
- The Makerspace has a collection of fasteners - check the cabinets or ask a helper.
- For 3D printed parts use $0.10 per gram for cost.
- Sparkfun.com can provide costs for many routine components.

Appendix B - Tunnel of Love Project Description

Overview:

Spongebob wants to take Pearl (his girl) to the fair. He hopes that the Tunnel of Love ride will warm their relationship. However, Squidward wants to chill Spongebob's love life. After some convincing, Spongebob gets Pearl into the ride. But, as the boat approaches the tunnel, Squidward appears with snowballs. This could be the end of Spongebob's plans! If you are a Squidward supporter, you must land a snowball on the love boat. If you are a Spongebob and Pearl supporter, help them arrive at the tunnel in warmth.

Getting Real:

In your teams, you need to produce a love boat (SB) and a ball thrower (SQ) using the following guidelines. During the competition you will compete against other teams.

Competition Setup:

1. The tunnel will have an entrance with a mouth 16" wide and 16" high. It tapers to a depth of 12" where it becomes 6" wide and 16" tall.
2. (SQ) The throwing platform will be a 16" by 16" platform at a height of 16" above the floor. The thrower cannot extend beyond the edges of the platform at any point.
3. (SQ) The team can use up to 4 ping-pong (snow) balls.
4. (SB) The boat will start somewhere (semi-random) at a radius of 3' from the center of the tunnel entrance.
5. (SQ) The thrower will be placed first, before the boat, and cannot be moved after the boat is placed.

Objectives:

1. (SB) To enter the tunnel in under 100 seconds without being hit, or hit as few times as possible.

    Score (lower is better) =
    + number of seconds
    + 50 * number of hits
    - 50 for reaching the 12" entrance
    - 50 for reaching the 6" entrance

2. (SQ) To hit SB with all of the balls.

    Score (higher is better) =
    + first hit * (100 - seconds)
    + second hit * (100 - seconds)
    + third hit * (100 - seconds)
    + fourth hit * (100 - seconds)

Constraints:

1. (All) The devices must be self-contained with no external wires. This means that it must be battery-powered and only communicate wirelessly.
2. (All) Any of the materials in the parts kit may be used. Materials are also available in the Makerspace. 3D printing is encouraged. Purchasing is discouraged but not prohibited.
3. (All) Intentional damage is not allowed.

Submissions:

Mon. Nov. 7 - Design Concepts
Mon. Nov. 21 - Competition Dry Run
Tues. Dec. 13 - Competition

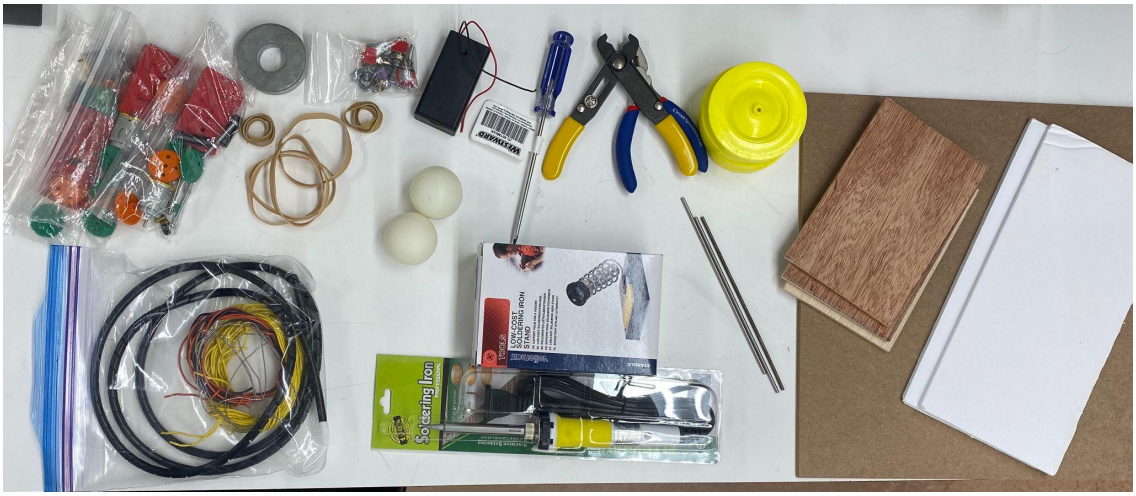
Tues. Dec. 13 - Documentation is complete in the shared document. Each team member must upload a link to the document.

Documentation Requirements:

This list will be increased as the project evolves. Note that ALL of the components must be combined into a single word, google doc, or PDF file.

1. Cover page including date, names, course, instructor
2. Project Concepts (as images)
3. WBS and Gantt Chart
4. Requirements Matrix
5. Project Description - 1 Per Team Member
6. Parts Spreadsheet
7. CAD Files
8. Pictures of purchased components
9. Pictures of the assembled devices. These should have enough detail to see the operation, inside, and outside.
10. Programs - make sure you comment all programs. This may include HTML files.
11. Pictures of the circuits.
12. Other items not listed above but necessary to describe the design.

Resources:



Gearbox Datasheet - https://drive.google.com/file/d/1MSR1-6CBNvUNvNUTjxMcmAaIvouMtgFo/view?usp=share_link

Ping Sensor - https://www.instructables.com/Interfacing-Ultrasonic-Sensor-Using-Raspberry-Pi-P

Appendix C - The Full Python Program for the Pico

```
# this program will do the following:

# 1. create an accesspoint - users will be able to login using a wifi
connection
#    using the network name and credentials below.
ssid = "idstealer" # the name of the network. Should be different than the
other wifi names
password = "012345678" # needs to be at least 8 characters


# 2. provide a simple webserver that will return the contents of the index.html
file
#    the address (probably 192.168.4.1 - see the message below) should be typed
into
#    the address line in the web browser (after you connect to the access
point)


# Notes of interest:
# 1. This program contains many print statements. These are for debugging only.
# 2. Functions are defined first. The main program steps are at the end of this
file.
# 3. The SSID/ESSID is the wifi network name
# 4. Computers on the internet can opt to use ports from 0-65535 (2^16)
# 5. The web server usually uses socket 80 for unencrypted incoming traffic
# 6. Try 192.168.4.1/bubba.html



# Author: H.Jack
# Last Modified: Sept 26, 2022


import socket
import network
import time
from machine import Pin, PWM

servopin = Pin( 1 )
pwm = PWM( servopin ) # servo signal hooker to GPIO 0
pwm.freq( 50 ) # this gets the 20 ms period needed by the servo motor

# this function is called to get the access point started.
def setup_access_point(ssid, password):
    print('setting up the device as an access point')
    ap = network.WLAN( network.AP_IF ) # We want a wireless network access
point
    ap.config( essid=ssid, password=password ) # the unique network stuff
    ap.active(True)

    while ap.active == False:
        pass # this does nothing, but it adds something inside the while loop
```

```
        # oddly, using anything else inside the loop leads to problems
    print("Yay: Access point active")

    config = ap.ifconfig() # get some data
    server_address = str(config[0])
    # people should be able to connect to the ssid now with the password
    print("Connect using IP address " + server_address ) # let the user know
our IP address
    # print(ap) # we could dump more data just for fun



def setup_webserver():
    # this sets up port 80 (http) to listen for incoming requests
    s = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
    s.setsockopt( socket.SOL_SOCKET, socket.SO_REUSEADDR, 1 ) # this fixes an
error EADDRINUSE
    s.bind( ('', 80) ) # this is the port (socket) for http/web
    s.listen( 5 )

    print('socket set up and we are now ready for web connections')
    return s


def get_page(filename):
    if (filename == '/') or (filename == '/index.html'): # this means a general
web page request
        actual_file = 'index.html'
    else:
        actual_file = 'error.html'

    with open(actual_file, 'r') as file:
        html_contents = file.read()
    return html_contents


# e.g. request is 'GET \'
def request_decode(request_string):
    request_list = request_string.split(' ')
    print('starts with _' + request_list[0] + '_' )
    if request_list[0] == 'GET':
        filename = request_list[1]
        print('got a GET command for file = _' + filename + '_')
        arguments = filename.split('?') # this will let us get arguments
        if len(arguments) == 1: #we have a simple file request
            response_string = get_page(filename)
        elif len(arguments) < 1: # oh no, something is very wrong
            response_string = 'Why did you send me broken HTML'
        else:
            variables = arguments[1].split('&')
            for variable in variables:
                variable = variable.split('=')
                print(variable[0])
                response_string = variable[1]
                if( variable[0] == 'angle' ):
```

```
                    print('we got a new angle ' + str( variable[1] ) )
                    pwm.duty_u16( int( variable[1] ) )
    else:
        print('Ouch, I do not know what to do with this')
        response_string = 'Nope'
    return response_string


def server_loop(s):
    print('Waiting for a new connection')
    conn, addr = s.accept()
    print('Got a connection from %s' % str(addr))
    request = conn.recv(2048) # this grabs up to 1024 characters
    #data_stuff = conn.read(1024)
    #print( data_stuff )

    print('      ')
    print('The following came from the remote web browser')
    #print('Content = %s' % str(request))
    request_string = request.decode() # this converts the characters to a
string
    print(request_string)
    response_string = request_decode(request_string) # if request is 'GET
index.html' we should get the file back

    print('   ')
    print( 'we are returning ' + response_string)
    conn.send('HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n') # Web
browsers like to see this
    conn.send(response_string)
    conn.close()
    print('    ')


######################## The big show

setup_access_point( ssid, password )
s = setup_webserver()
while True:
    server_loop(s)
```

Appendix D - index.html File

```
<html>
<body>
<p>It's alive
<h2>HTML Form</h2>
<form action="/index.html" method="GET">
  <label for="SSID">Network Name (SSID):</label><br>
  <input type="SSID" id="SSID" name="SSID"
value="wifi_network_name"><br>
  <label for="password">Password:</label><br>
  <input type="password" id="password" name="password"
value=""><br><br>
  <label for="angle">Servo Angle (2000-20000):</label><br>
  <input type="text" id="angle" name="angle" value=""><br><br>

  <input type="submit" value="Submit">
</form>
<p>If you click the "Submit" button, the form-data will be sent.</p>
<ul>
    <LI> one <b> And some bold text </b> and <i> italic </i>
    <li> two
</ul>
</body>
</html>
```

Appendix E - error.html file

```
<html>
<body>
<h1>Oops, something went wrong</h1>
</body>
</html>
```