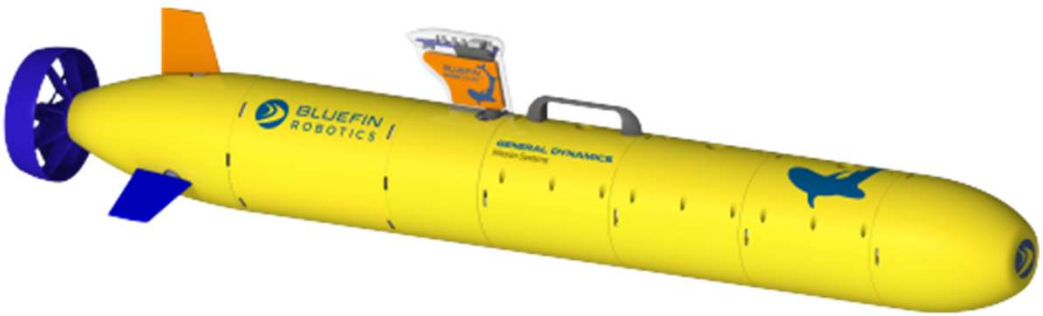
Autonomous Underwater Vehicle Status Update (9/25/2023)

William Buhrig, Nick Eoff, Tom Herlihy, Matt Laverty, Justin Day

Outlook on Semester Goals

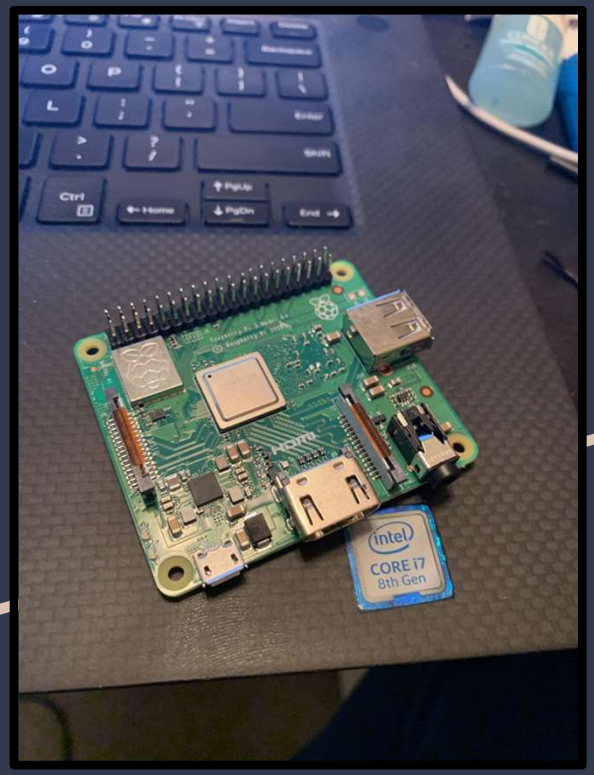
- Waterproofing
- Raspberry Pi Coding
- Buoyancy
- Remote transmission



Waterproofing

- In our first round of testing it was found that the O-ring on the top part of the AUV was leaking water.
- Install thicker O-ring.
- Apply lubricant to improve water resistance.

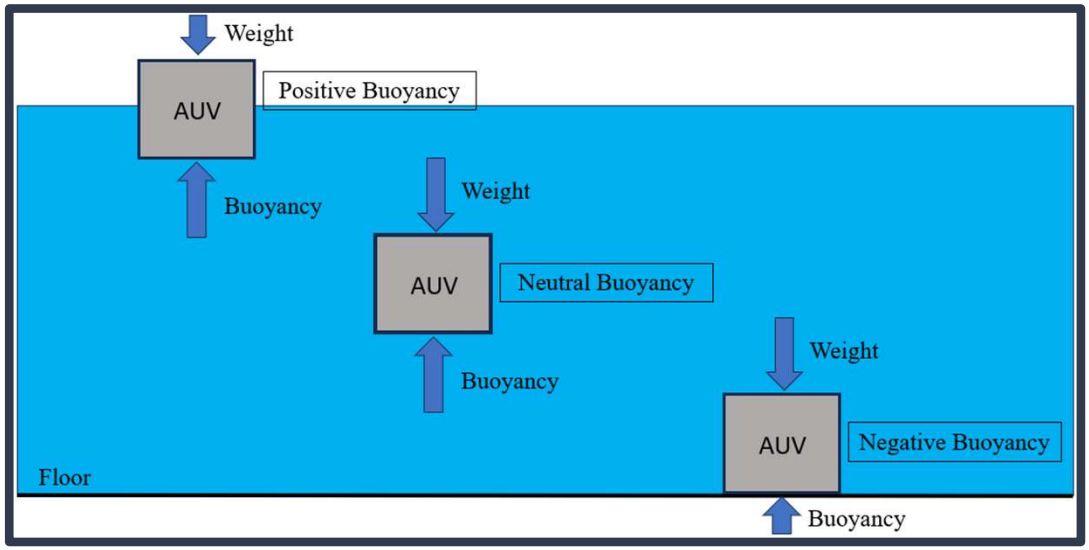
Raspberry Pi Coding




```
void setup() {
  // Open serial connection to computer
  Serial.begin(9600);
}

void loop() {
  // Read analog value from pin A0
  int value = analogRead(A0);

  // Scale value to range 0-255
  int scaled_value = map(value, 0, 1023, 0, 255);

  // Send scaled value over serial connection
  Serial.println(scaled_value);

  delay(100);
}
[11:32 PM]
import spidev
import serial

# Open SPI connection to MCP3008 chip
spi = spidev.SpiDev()
spi.open(0, 0)

# Open serial connection to computer
ser = serial.Serial('/dev/ttyS0', 9600, parity=serial.PARITY_NONE, stopbits=serial.STOPBITS_ONE)

# Main loop
while True:
    # Read analog value from MCP3008 chip
    adc_value = spi.xfer2([1, 8 + 0 << 4, 0])
    value = ((adc_value[1] & 3) << 8) + adc_value[2]

    # Scale value to range 0-255
    scaled_value = int(value / 1023.0 * 255)

    # Send scaled value over serial connection
    ser.write(str(scaled_value).encode() + b'\n')
[11:32 PM]
import serial
import RPi.GPIO as GPIO

# Set up GPIO pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)

# Open serial connection to computer
ser = serial.Serial('/dev/ttyS0', 9600, parity=serial.PARITY_NONE, stopbits=serial.STOPBITS_ONE)

# Main loop
while True:
    # Read command from serial connection
    command = ser.readline().decode('utf-8').strip()

    # Take action based on command
    if command == 'GPIO18_ON':
        GPIO.output(18, GPIO.HIGH)
    elif command == 'GPIO18_OFF':
        GPIO.output(18, GPIO.LOW)
    elif command == 'GPIO23_ON':
        GPIO.output(23, GPIO.HIGH)
    elif command == 'GPIO23_OFF':
        GPIO.output(23, GPIO.LOW)
    elif command == 'GPIO24_ON':
        GPIO.output(24, GPIO.HIGH)
    elif command == 'GPIO24_OFF':
        GPIO.output(24, GPIO.LOW)
[11:32 PM]
import serial
import pygame

# Initialize Pygame joystick module
pygame.init()
pygame.joystick.init()

# Open serial connection to Raspberry Pi
ser = serial.Serial('/dev/ttyS0', 9600, parity=serial.PARITY_NONE, stopbits=serial.STOPBITS_ONE)

# Main loop
while True:
    # Process Pygame events
    for event in pygame.event.get():
        if event.type == pygame.JOYAXISMOTION:
            # Map joystick input to command to send to Raspberry Pi
            if event.axis == 0:
                # Joystick moved left/right, send command to GPIO pin 18
                if event.value < -0.5:
                    ser.write(b'GPIO18_ON\n')
                elif event.value > 0.5:
                    ser.write(b'GPIO18_OFF\n')
            elif event.axis == 1:
                # Joystick moved up/down, send command to GPIO pin 23
                if event.value < -0.5:
                    ser.write(b'GPIO23_ON\n')
                elif event.value > 0.5:
                    ser.write(b'GPIO23_OFF\n')
        elif event.type == pygame.JOYBUTTONDOWN:
            # Joystick button pressed, send command to GPIO pin 24
            ser.write(b'GPIO24_ON\n')
        elif event.type == pygame.JOYBUTTONUP:
            # Joystick button released, send command to GPIO pin 24
            ser.write(b'GPIO24_OFF\n')
[11:32 PM]
import serial
import RPi.GPIO as GPIO

# Set up the GPIO pin
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

# Set up the serial connection
ser = serial.Serial('/dev/ttyS0', 115200)

# Listen for incoming data on the serial port
while True:
    data = ser.readline().decode('utf-8').rstrip()
    if data:
        # If data is received, toggle the GPIO pin
        GPIO.output(18, not GPIO.input(18))
```

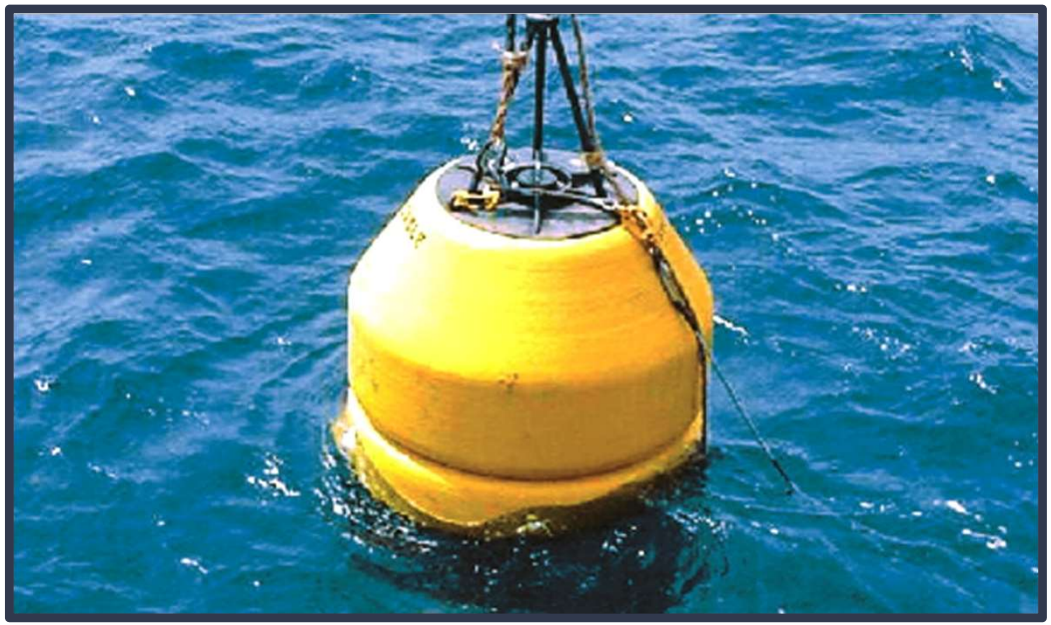
Buoyancy

- Add weight to the inside of the UAV to equalize the weight and the buoyancy force.



Remote Transmission

- The team will construct a buoy housing a transmitter that can be connected to a surface control system.
- The buoy will be tethered to the AUV.



Budget Update

- We were able to find a more affordable Raspberry Pi saving around $100.

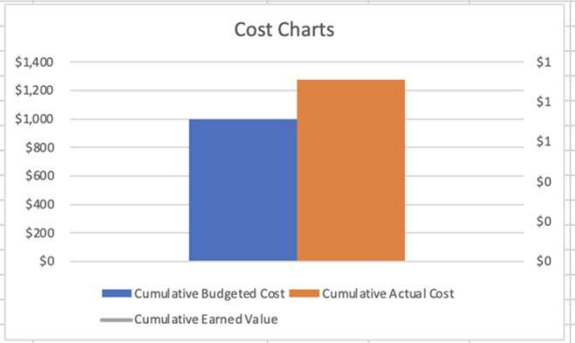
Wage Costs (so far)			
Project Members:	Hours Worked	Cost per Hour	Total Cost
Nick	8	$25	$200
Justin	8	$25	$200
Matt	8	$25	$200
Will	11	$25	$275
Colin	8	$25	$200
Tom	8	$25	$200
		Total Cost:	$1,275

Matierals Cost (to be purchased)	
Raspberry Pi	$50
Weights	$30
O-rings	$20
Transmitter and Reciever	$10
Total Cost:	$110

Equipment and Software Cost	
Python	$0
Auto CAD	$0
Laptops	$0
Excell	$0
Total Cost:	$0

Facilities Cost	
Dr. Kaipa's Pool	$0
Dr. Kaipa's Lab	$0
Total Cost:	$0

Total Budgeted Cost	$1,385	
Cumulative Budgeted Cost	$1,000	
Cumulative Actual Cost	$1,275	
Cumulative Earned Value	0% completed therefore no earned valu	$0
Cost Performance Index	1.08627451	
Cost Variance	7.94223827	



GANTT Chart

ID	Task Mode	Task Name	Duration	Start
1		Initial Team Meetings	12 days	Mon 8/28/23
2		Intial Advisor Meetings	6 days	Wed 9/13/23
3		Gather Materials	21 days	Wed 9/13/23
4		Prepare and Present Oral Presentation 1	6 days	Mon 9/18/23
5		Research and Implement Engineering Standards	11 days	Mon 9/18/23
6		Begin Design Improvements	21 days	Mon 9/25/23
7		Prepare and Present Mid Term Oral Presentation	6 days	Mon 10/9/23
8		Test Design	10 days	Mon 10/23/23
9		Prepare and Present Oral Presentation 2	6 days	Mon 10/30/23
10		Iterate Design Improvements	12 days	Sat 11/4/23
11		Final Design Testing	10 days	Mon 11/20/23
12		Prepare and Deliver Final Presentation and Poster	6 days	Wed 11/29/23
13		Prepare and Deliver Final Product	4 days	Fri 12/1/23