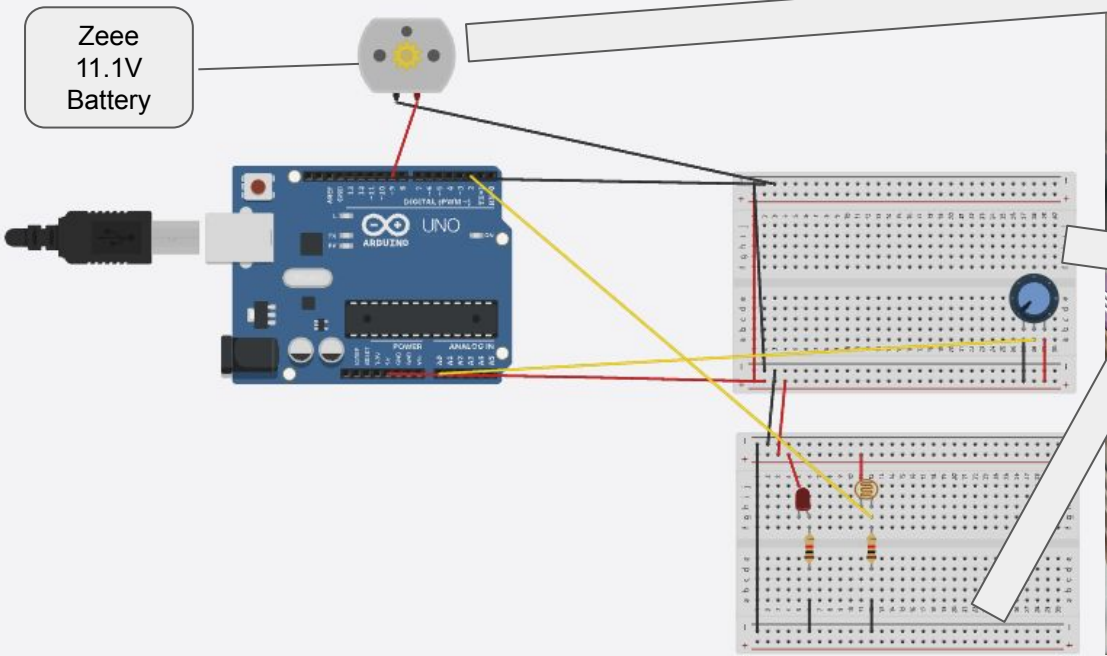
Propeller Speed Measurement

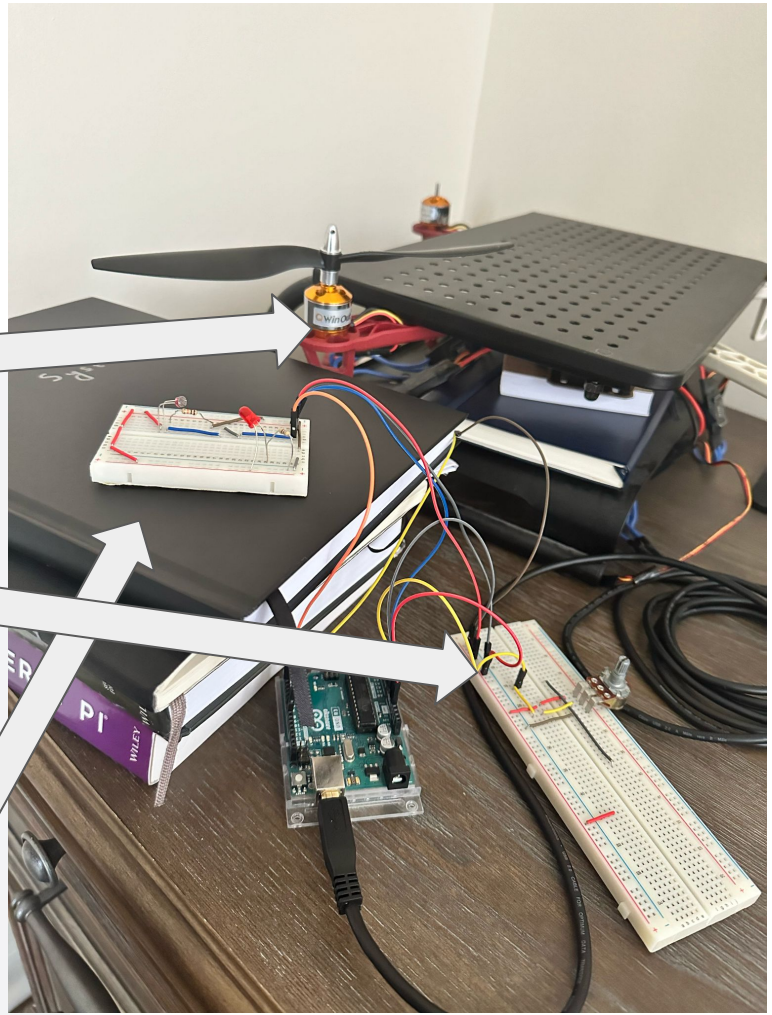
Julian Torres

Requirements

- Measure the speed of brushless propeller driven by an Arduino
- Measure using an IR Emitter/Detector pair
- Capture RPMs over time
- Graph RPMs over time
- Use Round Robin with Interrupts or Function Queue Scheduling

Circuit Design



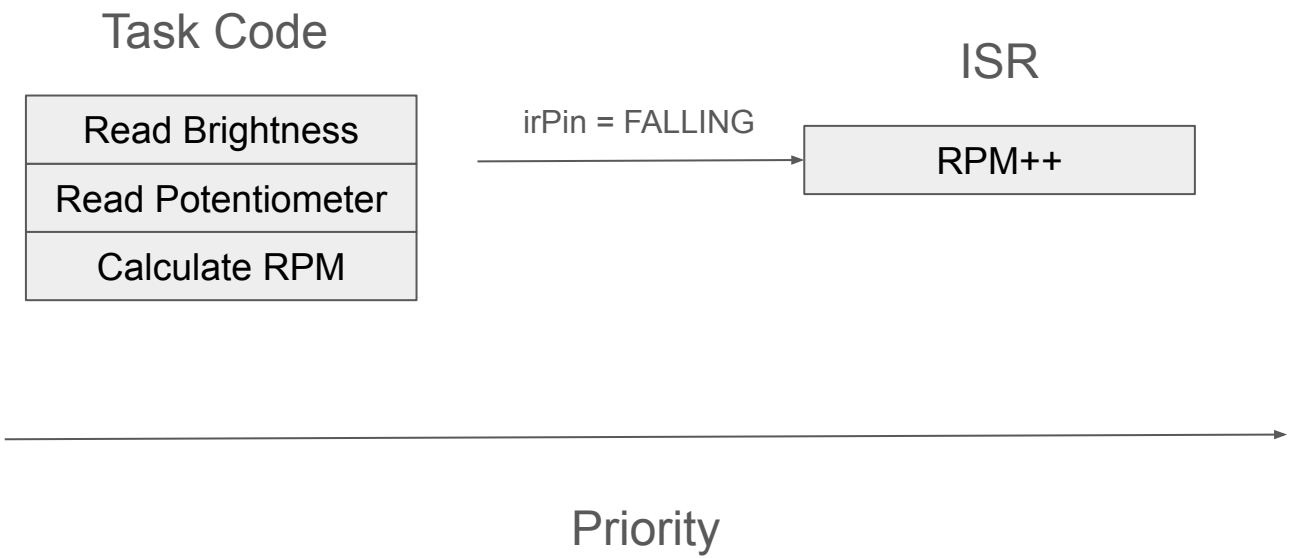


Execution

- LED facing Photoresistor measuring light input
- Blade (attached to motor) positioning to sweep between the two components
- Potentiometer used to modulate motor speed
- Arduino code used to compute measured RPM and sent via Serial Monitor to CoolTerm and written to a .csv file
- Plotted via simple Python script
- Test Programs:
  - Confirming Potentiometer functionality
  - Confirming Photoresistor functionality

CoolTerm: https://coolterm.en.lo4d.com/windows

Software Design - Round Robin with Interrupts



Code

- Global Variables
- Setup function
- ISR

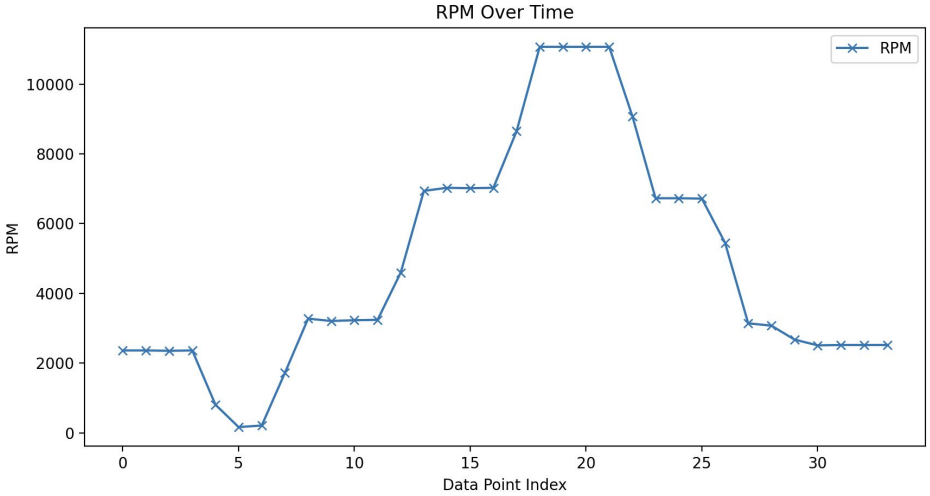
```
#include <Servo.h>
#include <TimerOne.h>

// Motor vars
Servo ESC;                      // Servo object for ESC

// Potentiometer vars
uint8_t potPin = A0;            // Analog pin location for potentiometer
int potValue;                   // Reading from potentiometer

// Photoresistor vars
int PhotoReaderPin_a = A1;      // Pin for photoresistor
int measuredLight;              // Analog read from photoresistor

// Interrupt and RPM vars
volatile int rpmCount = 0;      // Count of light interruptions
volatile long timeold = 0;      // Previous time for RPM calculation

volatile const int irPin = 2;   // Pin for photoresistor interrupt
volatile int rpm = 0;           // Measured RPM

long debugTime = 0;

// Enum to represent tasks in the round-robin scheduler
enum TaskState { READ_POT, READ_LIGHT, CALC_RPM };
TaskState currentTask = READ_POT;

// Interrupt Service Routine (ISR) for counting RPM
void rpmISR() {
  rpmCount++;
}

void setup() {
  Serial.begin(9600);

  // Set up ESC
  int pin = 9;
  int minPulsWidth = 1000;
  int maxPulsWidth = 2000;
  ESC.attach(pin, minPulsWidth, maxPulsWidth);  // Attach the ESC

  // Set up interrupt on irPin
  attachInterrupt(digitalPinToInterrupt(irPin), rpmISR, FALLING);

  // Initialize timing variables
  timeold = millis();
  debugTime = timeold;
}
```

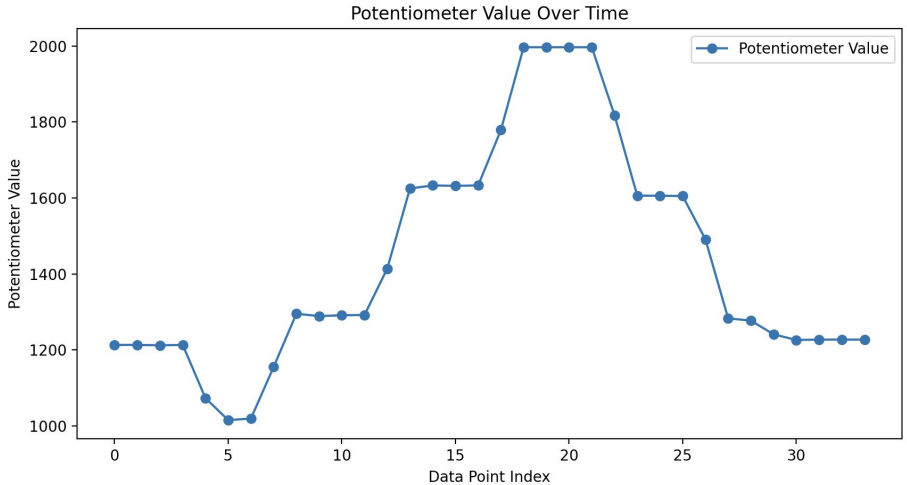
Code

Loop function

```
50  void loop() {
51    // Round-robin task scheduling
52    switch (currentTask) {
53
54      case READ_POT:
55        // Potentiometer task
56        potValue = analogRead(potPin);
57        potValue = map(potValue, 0, 1023, 1000, 2000);  // Map to ESC PWM range
58        ESC.write(potValue);                            // Set motor speed based on potValue
59        currentTask = READ_LIGHT;
60        break;
61
62      case READ_LIGHT:
63        // Light sensor task
64        measuredLight = analogRead(PhotoReaderPin_a);
65        if (millis() - debugTime >= 100) {
66          Serial.print("measuredLight,");
67          Serial.println(measuredLight);
68          debugTime = millis();                         // Reset debug timer
69        }
70        currentTask = CALC_RPM;
71        break;
72
73      case CALC_RPM:
74        // RPM calculation task
75        if (millis() - timeold >= 1000) {
76          // Calculate RPM: (rpmCount * 60) / Number of blades (assuming 1 blade)
77          rpm = abs((rpmCount * 60)) / 2;  // Multiply by 60 to convert to RPM (/2 for 2 blades)
78          Serial.print("RPM,");
79          Serial.println(rpm);
80
81          // Reset for next interval
82          timeold = millis();
83          rpmCount = 0;
84        }
85        currentTask = READ_POT;
86        break;
87    }
88  }
```

Results





Video: https://youtu.be/M3uefzmVEEE