

GNG 1103: Deliverable G Group D12

Introduction

The following document outlines additional prototypes created for the JAMZ drone design project. The secondary prototypes include additional coding, that tests a new spam prevention function. This will ensure that the drone operator is not overloaded with unnecessary data. Communication between the two sensors and the microprocessor, and communication between the microprocessor and the drone is the most critical subsystem of the design. Also included will be plans for future tests to ensure that the climate sensor works as expected.

Prototype

Testing Spam Prevention Function

```
wt = 5;
wh = 80;
if(Time>10000){
if (wt>65||wt<10)
  if((unsigned long)(Time-countT)>5000){
    Serial.println((String)"Temperature not ideal: Current temperature = "+wt+"C");
    countT = Time;
  if(wh>55)
  {
  if((unsigned long)(Time-countH)>5000){
    Serial.println((String)"Humidity not ideal: Current humidity = "+wh+"%");
    countH = Time;
  }
  if((unsigned long)(Time-test)>1000)
  {
   Serial.println((String)"Time: "+(Time/1000)+"s");
  test = Time;
```

Figure 1.1. Arduino spam prevention code¹

```
Temperature/humidity sensor reading abnormal
Time is: 20
Time is: 21
Time is: 22
Time is: 23
Time is: 24
Time is: 25
Time is: 26
Time is: 27
Time is: 28
Time is: 29
Time is: 30
Time is: 31
Time is: 32
Time is: 33
Time is: 34
Time is: 35
Time is: 36
Time is: 37
Time is: 38
Time is: 39
Temperature/humidity sensor reading abnormal
```

Figure 1.2. Arduino serial output

Previously, the Arduino would spam the raspberry pi with warnings if it met one of the pre-set conditions until it no longer met that condition. Using millis function, the internal time is tracked so that the previous output does not send unless a certain time has past. As shown in figure 1.2, the delay works as intended, sending only a signal after a designated period of time (in this case 20). This time data will not be sent to the raspberry pi for the actual code.

```
#include <Wire.h>
bool boxOpen=0;
unsigned long Time=0;
unsigned long closedTime=0;
char message;
bool newData;
void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600);
 pinMode(5,OUTPUT);
void loop() {
 // put your main code here, to run repeatedly:
void serialEvent(){
 while (Serial.available()) {
   message = Serial.read();
  if(message=='c')
    digitalWrite(5,LOW);
  else if(message=='o'){
   digitalWrite(5, HIGH);
   }
  else{
   Serial.println(" Input error");
  }
}
```

Figure 2.1. Arduino two way serial communication code²

```
import serial
import time
ser = serial.Serial('/dev/ttyUSBO',9600,timeout=1)
ser.flush()
time.sleep(2)
ser.write(b"c")
line = ser.readline().decode('utf-8').rstrip()
print(line)
```

Figure 2.2. Raspberry python script for sending and receiving serial data³

```
pi@webber:~/Documents/serial $ python sendO.py
pi@webber:~/Documents/serial $ python sendC.py
pi@webber:~/Documents/serial $ python sendX.py
Input error
pi@webber:~/Documents/serial $
```

Figure 2.3. Raspberry pi script calls

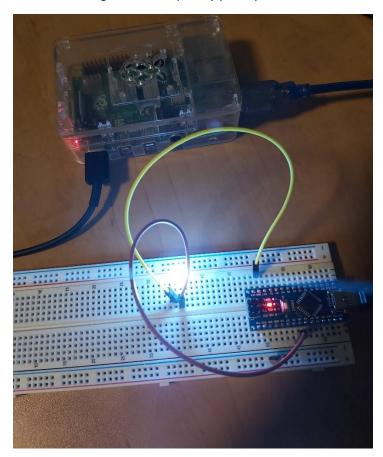


Figure 2.4. Arduino output

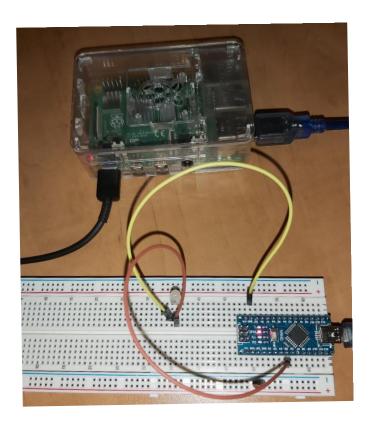


Figure 2.5. Arduino output

This prototype was to make sure that the Arduino can receive information from the raspberry pi, understand and store that information, use the information to alter other variables within the Arduino. This is needed as the Arduino needs to know if the box is open to not send false positive warnings to the operator. The serialEvent is unique as it doesn't have to be called in the loop but rather acts as an interrupt. However, the serialEvent is not a true interrupt since it is run after each loop instead of pausing the moment a message comes in. As shown in the prototype demo, this test was successful, having the two-way Arduino and raspberry pi communication with the input from raspberry pi altering a condition in Arduino high/low, and Arduino transmitting data back to the Pi. Like the previous prototype, this will have to be tested with the GPIO pin to make sure that it will work on the drone, but for now this prototype proves that the code is functional and works as intended.

With all the important aspects of the code tested, the 'final' code was created, implementing all the techniques and functions from prototype I and the ones shown above. This will have to be tested with the real sensor using GPIO pins. The 'final' code with the details can be found attached to the submission. This code will have to be optimized and tweaked most likely due to the different configuration that may occur once it is tested with the GPIO.

Future Test Plans

Next steps are to create a comprehensive analytical model of all the subsystems. This will be used to test size, weight and confirm placement of the different components relative to the drone. Cases will be printed, using a 3D printer, for the Arduino, the sensors and the voltage converter. The Arduino case will be tested to ensure that water does not enter the case through the holes where wires exit. It will also be tested for impact resistance; in case the drone was to crash. Being impact resistant will allow parts to be salvaged from the drone. A virtual wind tunnel testing will also be considered to test for drag force. After the last client meeting on Friday, March 12, the aerodynamic criterion of the Arduino case was considered. Another design will be created, and a comparison will be done between the original and the aerodynamic model for all Arduino case testings. The case for the sensor is used to protect the sensor from the climate inside the box. It will be tested to ensure that the sensor will function properly while encased. When the sensors come in, the code will be tested to ensure that it works as intended.

Conclusion

This purpose of this prototype was to make sure that the Arduino can receive information from the raspberry pi, understand and store that information, use the information to alter other variables within the Arduino. These features are necessary since the Arduino has to be informed if the box is open to not send false positive warnings to the operator as this causes confusion and misinformation. As shown in the prototype demo, it was a successful test. The important aspects of the code were also tested, and the final code was created, implementing all the techniques and functions from prototype I and shown in this report. This will have to be tested with the real sensor using GPIO pins in the next prototype as discussed earlier.

References

- 1. https://www.baldengineer.com/arduino-how-do-you-reset-millis.html
- 2. https://www.arduino.cc/en/Tutorial/BuiltInExamples/SerialEvent
- 3. https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/