Deliverable F - Prototype I and Customer Feedback

GNG 1103D

Group #9

March 9, 2021

Group members: Karen Hakko, Elsa Lange, Tri Thai, Jacob Troop and Sandeep Sinha

Contents

List of Figures

List of Tables

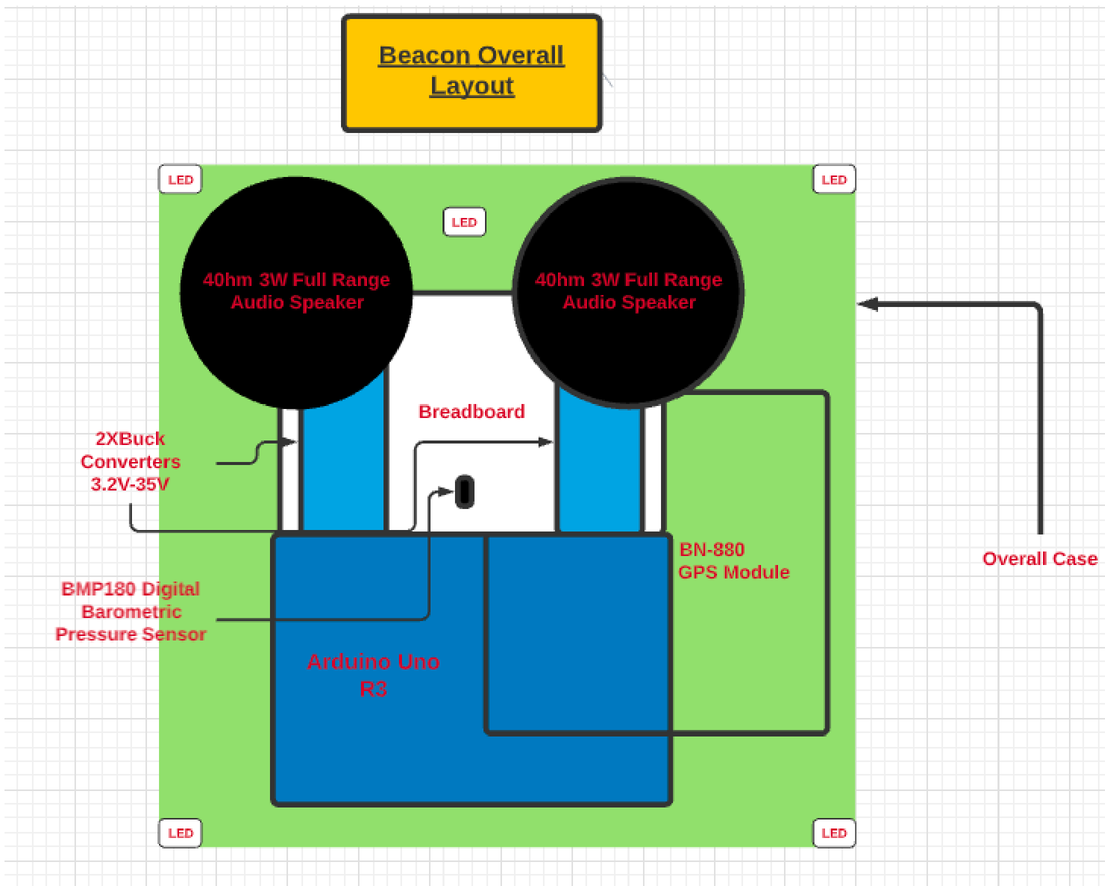
Problem Statement

The JAMZ developers need an emergency beacon that transmits accurate and quick location information about the drone to the operator in live time by interpreting the data received from the sensors as well as alerting nearby citizens of the downed drone with flashing lights and a voice system.

Summary of Previous Deliverable

In the previous deliverable we created a bill of materials (BOM) to ensure we had all the required components and stayed within the budget set by JAMZ. After the BOM was completed, a detailed global concept that includes all the parts from the BOM was created to make a visual plan of what the overall system should look like.

Figure 1. Overall Beacon Layout



Task plans were then created for the three prototypes which included a detailed list of all required tasks along with the member responsible and the estimated date of completion for each task. A risk assessment followed each of the task plans to help our group better understand the potential problems we may face while creating the prototypes.

Table 1 - Probability of Potential Risks Rating System

Legend	Orange	Red
Probability	Unlikely.	Reasonably likely.

The risks were assessed based on the probability that they would occur and contingency plans were created to refer to if our group runs into any issues while creating the prototypes. The previous deliverable also included a list of required materials for each of the prototypes and as always a Wrike update.

Prototype I

Table 2 - Task Plan for Prototype I

Task		Member Responsible	Due date
Research	1. Arduino and sensor a. Electronic component and wiring b. Test plan (code) 2. Arduino and GPS a. Electronic component and wiring b. Test plan (code) 3. Arduino and voice system a. Electronic component and wiring b. Test plan (code) 4. Arduino and light system	Elsa Sandeep Karen Jacob	March 2nd, 2021

	a. Electronic component and wiring b. Test plan (code) 5. All-around connections a. Electronic component and wiring b. Test plan (code)	Tri	
Assembly	1. Arduino and sensor a. Electronic component and wiring 2. Arduino and GPS a. Electronic component and wiring 3. Arduino and voice system a. Electronic component and wiring 4. Arduino and light system a. Electronic component and wiring 5. All-around connections a. Make sure all components are connected	Elsa Sandeep Karen Jacob Tri	March 4th, 2021

Deliverable F	1. Prototyping test plan a. Test on Tinkercad i. Altitude subsystem ii. Location subsystem iii. Voice system iv. Light system 2.Formatting 3. Presentation 4. Wrike update	 Elsa Sandeep Karen Jacob Jacob Tri Karen	March 5th, 2021

The purpose of this prototype and its several subsystems was to visually represent the assembly of the electrical components and test the subsystems both in an isolated and fully assembled circuit. It was assumed throughout the deliverable that the Tinkercad simulations were a close representation of the real-life tests. The results of this deliverable will be used in the second prototype as a reference for the physical design.

Altitude Subsystem

This section contains testing done for the altitude subsystem in prototype I. Tinkercad was used as an initial prototype to decrease the amount of mistakes done on the first physical prototype. Moreover, Tinkercad could help us plan how to assemble the physical prototype.

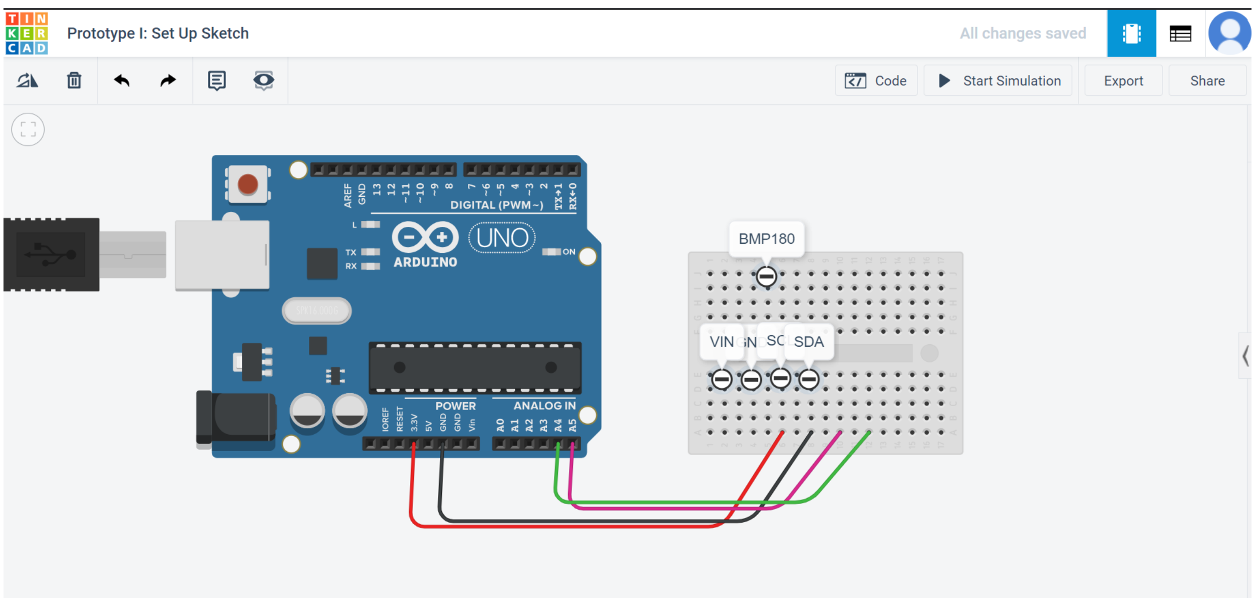
Table 3 - Test Plan 1 for Altitude Subsystem of Prototype I

Test ID	Member Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
1	Elsa	To plan the assembly of the electrical components of the Arduino and the BMP180 barometric	There is no way to test the set up as the BMP180 is not available on Tinkercad. The prototype was a visual representation	The sketch provided a visual representation of how the pressure sensor should be connected to the Arduino R3. These results were used as a reference to	March 4th, 2021 Test Duration: n/a.	The prototype was deemed satisfactory when it incorporated all

		pressure sensor.	of the isolated altitude subsystem.	create the full assembly model on Tinkercad and will be used in the future for the physical prototype.		the necessary compone nts that were represent ed in the Tinkerca d circuit.

This test was done to visually represent the altitude subsystem to facilitate its representation in the overall subsystem. There was no real "test" in this prototype as the BMP180 could not be integrated in Tinkercad. The test was deemed satisfactory when all the components needed to set up the altitude subsystem were accounted for. Afterwards, the results in this test will be used to assemble the physical prototype including all of the subsystems.

Figure 2. Assembly of Altitude Subsystem



In the figure above, the plan of the assembly is shown. As the BMP180 could not be physically represented on Tinkercad, a reference point was used to represent it and other components. The figure shows the Arduino and the names of the components that would also be included in the altitude subsystem: the BMP180, the vin (where the voltage would come in from), the GND (the ground, where the voltage can be measured from), the SCL (the serial clock pin which is used has regular time intervals to synchronize data transmission) and the SDA (which will transmit the data). The wiring is also displayed in the circuit. This circuit design is an idealised version of the altitude subsystem as it does not include any of the other subsystems. The knowledge gained from the 4th lab was used to tailor this original design and the design in the figure can be found here.

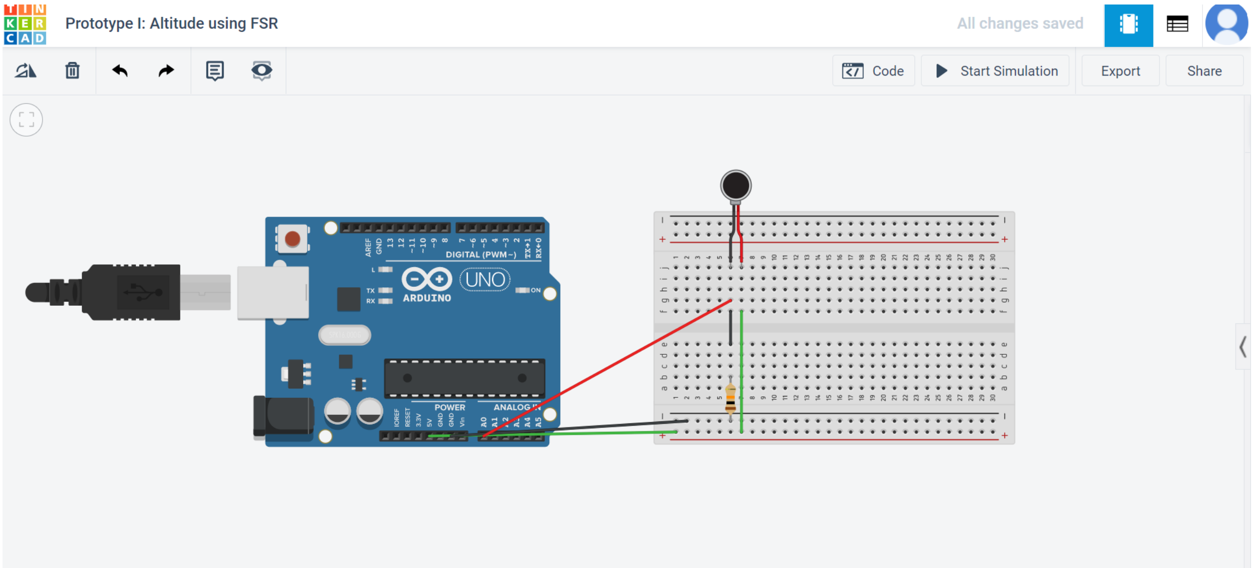
A second test for this subsystem was done to test a similar code to the code that will test the BMP180.

Table 4 - Test Plan 2 for Altitude Subsystem of Prototype I

Test ID	Member Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
2	Elsa	To test a similar test code for the BMP180.	The test method will be done on Tinkercad and will verify the execution of the test code.	These results increased the understandin g of Arduino codes as well as a baseline for the execution of pressure codes.	March 4th, 2021 Test Duration: Approximately 2 minutes.	The testing was deemed complete when the code executed consistent, live data.

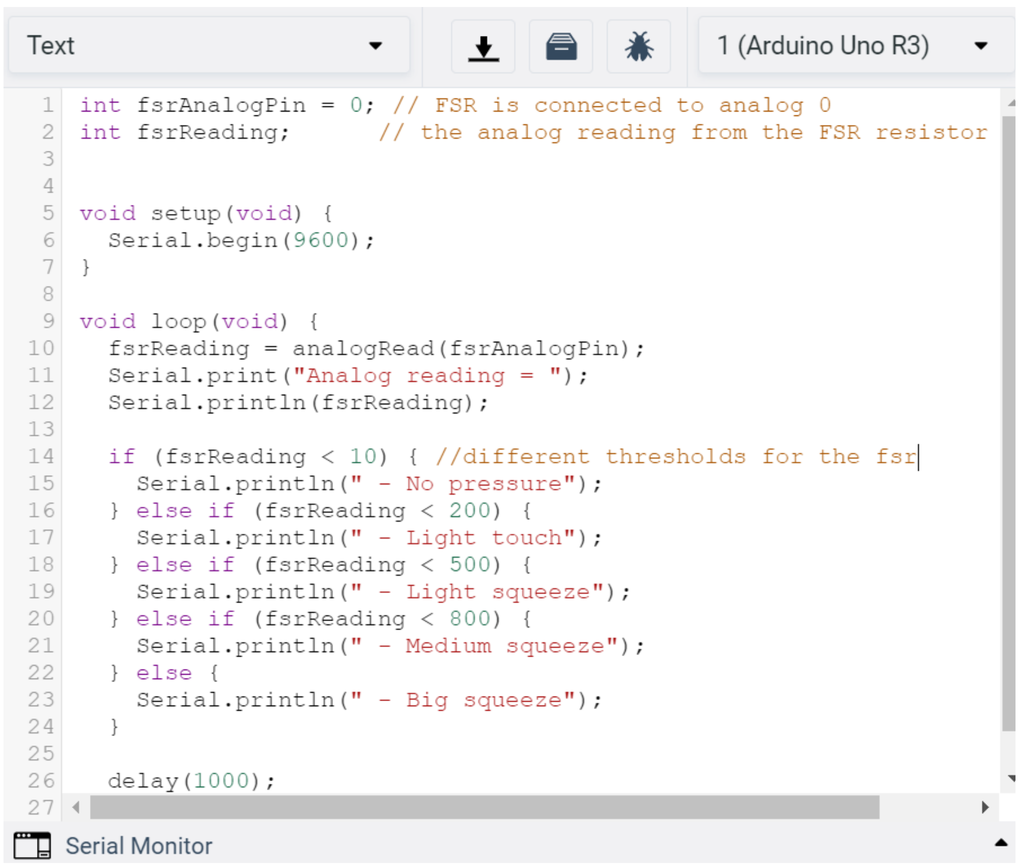
This test was done to test a similar code to the BMP180 as the actual product could not be tested on Tinkercad. Instead, a vibration sensor was used to mimic the reading of the information from the sensor and classify that information. The test was simulated on Tinkercad for about 2 minutes and was deemed complete as the execution of the code satisfied the design requirements of live, reliable data. These results will be used in the future to facilitate the use of test code for the BMP180.

Figure 3. Assembly of the Vibration Sensor



The figure above shows the Arduino, resistor and vibration sensor. The Tinkercad sketch was edited from this original design and the figure above can be found here.

Figure 4. Test Code for the Vibration Sensor




```
int fsrAnalogPin = 0; // FSR is connected to analog 0
int fsrReading;      // the analog reading from the FSR resistor


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

void loop(void) {
  fsrReading = analogRead(fsrAnalogPin);
  Serial.print("Analog reading = ");
  Serial.println(fsrReading);

  if (fsrReading < 10) { //different thresholds for the fsr
    Serial.println(" - No pressure");
  } else if (fsrReading < 200) {
    Serial.println(" - Light touch");
  } else if (fsrReading < 500) {
    Serial.println(" - Light squeeze");
  } else if (fsrReading < 800) {
    Serial.println(" - Medium squeeze");
  } else {
    Serial.println(" - Big squeeze");
  }

  delay(1000);
```

The figure above shows the code used to test the vibration sensor. The code reads the reading of the vibration sensor and the thresholds were added to classify the reading based on the input. The code was retrieved from this website.

Figure 5. Executions of the Test Code for Vibration Sensor

```
Serial Monitor
Analog reading = 1018
 - Big squeeze
Analog reading = 1018
 - Big squeeze
Analog reading = 1018
 - Big squeeze
Analog reading = 1018
 - Big squeeze
```

The figure above shows the execution of the code for this test. The output of this code was consistent (as there were no changes to the vibration sensor during the simulation) and was live as the execution occurred almost instantaneously.

Location Subsystem

The location subsystem for Prototype 1 is geared around having a circuit that can generate the location of the drone via a GPS sensor, and can calculate the latitude and longitude of the drone. Figure 1 shows the circuit of the GPS subsystem on TinkerCAD, and Figure 2 shows the code needed for the subsystem to function. The coordinates of the drone will be shown on two LCD displays. The first display will show the local time of the drone, and the second display will show the latitude and longitude of the drone.

Table 5 - GPS System Test Plan for Prototype I

Test ID	Member Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
1	Sandeep	To test whether the GPS module and code will display location	The circuit will be designed on TinkerCAD and the code will also be	The code worked well with the circuit, and the GPS sensor outputted	March 4th,2021 The test was run for 2 minutes and 30 seconds.	The test was run until the tester was certain that the data output was factually

		data.	inputted on TinkerCAD in order to test the reliability of the circuit.	the data required, the latitude and longitude, as well as the local time.		correct and consistent when the testing laptop was moved around.

Figure 6. The GPS subsystem circuit via Tinkercad

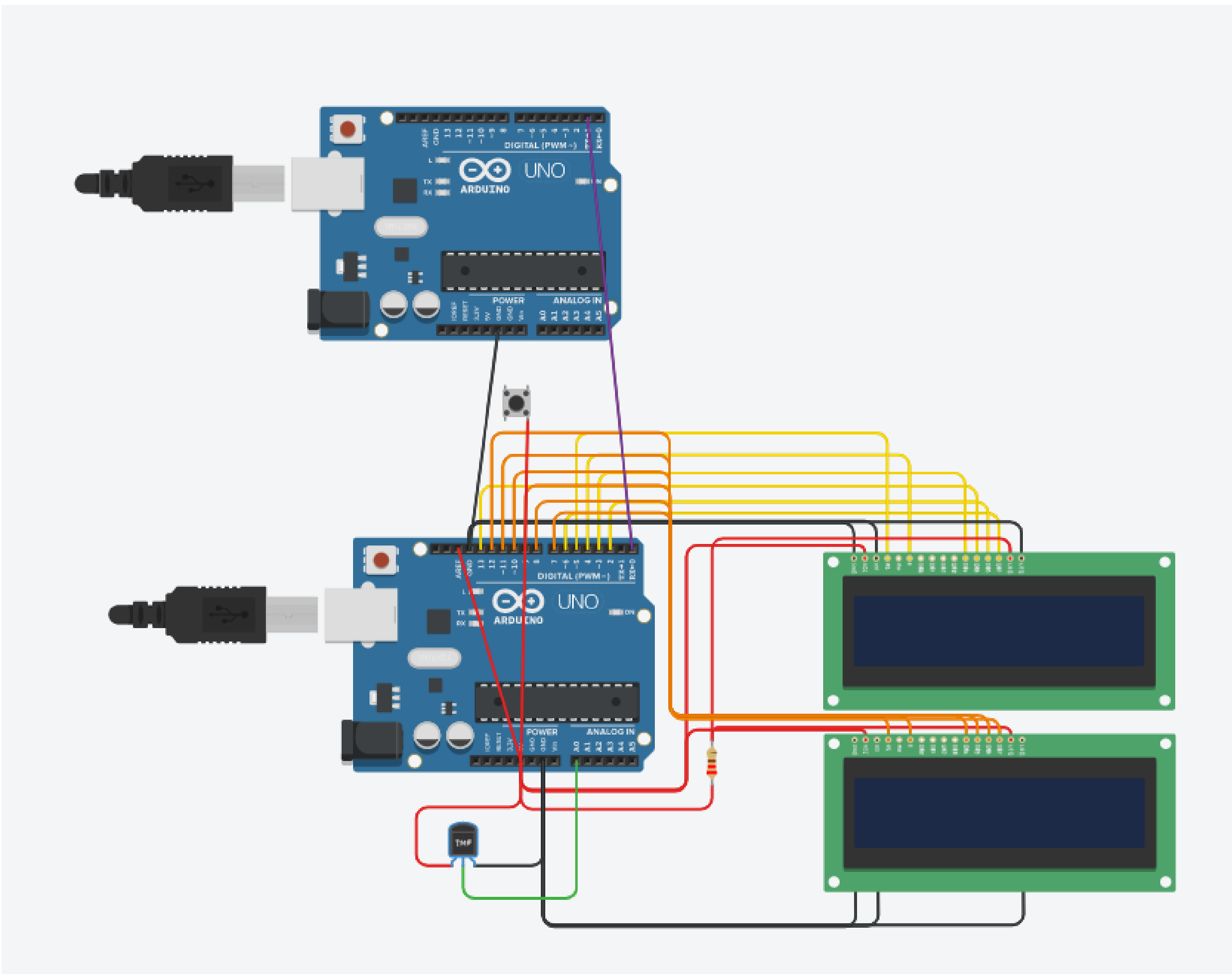


Figure 7. Part One through Seven of the testing code

```
#include <LiquidCrystal.h>

#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (0<<ADLAR))

float stepADC = 0.0048828125;

LiquidCrystal lcd1(5, 4, 3, 13, 2, 6);
LiquidCrystal lcd2(12, 11, 10, 9, 8, 7);

char sSecventa[119];

//declarate global deoarece in cazul calculatiilor din ecranul 2
//memoria este deja scrisa si ar fi trebuit o alta initializare
//cu mai multe varibile, etc..
static char *aValori[2];
static char *token;
static char *aValori1[9];
static char *token1;
static char *aValori2[15];
static char *token2;

int contor = 0;
bool ecran = 0;
bool stareButon = 0;
bool stareButonAnterioara = 0;
//--------------------

unsigned int citesteADC(unsigned int adc_input)
{
  ADMUX = adc_input | ADC_VREF_TYPE;

  //delay necesar pentru stabilirea ADC pt tens
  //de intrare analogica
  delayMicroseconds(10);

  //start conversie
```

```
  //delay necesar pentru stabilirea ADC pt tens
  //de intrare analogica
  delayMicroseconds(10);

  //start conversie
  ADCSRA |= (1<<ADSC);
  //Asteptare finalizare conversie
  while ((ADCSRA & (1<<ADIF))==0){}
  ADCSRA |= (1<<ADIF);

  //returnare rezultat pe 16 bitia
  return ADCW;
}

//cod recunoastere/segmentare secventa
void segmentareSecventa()
{
  //segmentare secventa principala
  token = strtok(sSecventa, "/");
  static int increment = 0;
  while (token != NULL)
  {
    aValori[increment++] = token;

    token = strtok(NULL, "/");
  }

  //segmentare secventa 1
  token1 = strtok(aValori[0], ",");
  static int increment1 = 0;
  while (token1 != NULL)
  {
    aValori1[increment1++] = token1;

    token1 = strtok(NULL, ",");
  }
```

```

  //segmentare secventa 2
  token2 = strtok(aValori[1], ",");
  static int increment2 = 0;
  while (token2 != NULL)
  {
    aValori2[increment2++] = token2;

    token2 = strtok(NULL, ",");
  }

  //incheiere segmentare secvente -------------
}

void ecranulUnu()
{
  //afisare viteza
  static float iViteza = 0;
  if (strcmp(aValori1[0], "$GPVTG") == 0)
  {
    iViteza = atof(aValori1[7]);
  }
    //setare cursor: coloana 0, linia 1
  lcd1.setCursor(0, 1);
  lcd1.print("Vit: ");
  lcd1.print(iViteza);
  lcd1.print("km/h");

  //afisare timp
  static long int temp[4];
  static int iSateliti = 0;
  if (strcmp(aValori2[0], "$GPGGA") == 0)
  {
    //stocare sir
    temp[0] = atof(aValori2[1]);
    //stocare secunde
```

```
    temp[3] = temp[0] % 100;
    temp[0] = temp[0] / 100;
    //stocare minute
    temp[2] = temp[0] % 100;
    temp[0] = temp[0] / 100;
    //stocare ore
    temp[1] = temp[0];

    iSateliti = atoi(aValori2[7]);
  }

  //setare cursor: coloana 0, linia 0
  lcd1.setCursor(0, 0);
  lcd1.print("Timp: ");
  lcd1.print(temp[1]);
  lcd1.print(":");
  lcd1.print(temp[2]);
  lcd1.print(":");
  lcd1.print(temp[3]);


  //setare cursor: coloana 0, linia 3
  lcd2.setCursor(0,0);
  lcd2.print("Nr. sat: ");
  lcd2.print(iSateliti);

  //citire termometru de pe pinul analogic "0"
  unsigned int sensorValue = citesteADC(0);
  //calculare temperatura in functie de valorile din datasheetul termometrului
  float fTemperature = (stepADC*sensorValue-0.5)*100;

  //setare cursor: coloana 0, linia 3
  lcd2.setCursor(0, 1);
  lcd2.print("Temp: ");
  lcd2.print(fTemperature, 2);
  lcd2.print(" C");
```

```
  //incheiere prelucrare ecran unu--------
}

//afisare/selectare date pe ecranul doi
void ecranulDoi()
{
  //segmentareSecventa();
  //calculare, afisare latitudine
  static float fLatitudine = 0;
  static float fLongitudine = 0;
  static float fAltitudine = 0;
  if (strcmp(aValori2[0], "$GPGGA") == 0)
  {
    fLatitudine = atof(aValori2[2]);
    fLongitudine = atof(aValori2[4]);
    fAltitudine = atof(aValori2[9]) - atof(aValori2[11]);
  }
  //afisare date
  //setare cursor: coloana 0, linia 1
  lcd1.setCursor(0, 1);
  lcd1.print("Lat: ");
  lcd1.print((int)fLatitudine/100+((int)(fLatitudine)%100+(fLatitudine-(int)(fLatitudine)))/60);
  lcd1.print(aValori2[3]);
  //setare cursor: coloana 0, linia 2
  lcd2.setCursor(0, 0);
  lcd2.print("Long: ");
  lcd2.print((int)fLongitudine/100+((int)(fLongitudine)%100+(fLongitudine-(int)(fLongitudine)))/60);
  lcd2.print(aValori2[5]);
  //setare cursor: coloana 0, linia 3
  lcd2.setCursor(0, 1);
  lcd2.print("Alt: ");
  lcd2.print(fAltitudine);
  lcd2.print("m");
  //incheiere prelucrare ecran doi--------
}
```

```
//--------------------
// -------SETUP--------
//--------------------
void setup()
{
  Serial.begin(9600);

  lcd1.begin(2,16);
  lcd2.begin(2,16);
}

//--------------------
// -------LOOP---------
//--------------------
void loop()
{
  if(contor < 1)
  {
    Serial.readBytes(sSecventa, 119);
    Serial.println(sSecventa);
    segmentareSecventa();
    contor++;
  }

  //citire stare buton
  /*stareButon = (PIND & (1 << PIND2));

  if(stareButon != stareButonAnterioara)
  {
    delay(200);
    ecran = !ecran;

    if( stareButon == 0)
    {
      stareButon = !stareButon;
    }
```

```
  {
    Serial.readBytes(sSecventa, 119);
    Serial.println(sSecventa);
    segmentareSecventa();
    contor++;
  }

  //citire stare buton
  /*stareButon = (PIND & (1 << PIND2));

  if(stareButon != stareButonAnterioara)
  {
    delay(200);
    ecran = !ecran;

    if( stareButon == 0)
    {
      stareButon = !stareButon;
    }
  }
  stareButonAnterioara = stareButon;

*/

    ecranulUnu();
    delay(1000);
    lcd1.clear();
    lcd2.clear();

    ecranulDoi();
    delay(1000);
    lcd1.clear();
    lcd2.clear();

}
```

The Tinkercad design shown in the figure can be found here.

Voice Subsystem

The voice system for prototype I is focusing on finding the right wiring to connect the speaker to an Arduino. Figure 1 shows the speaker setup and its connection to the Arduino based on the research done. Figure 2 displays the initial code used to test the correctness of the wiring connections. This initial test does not play the automated message which notifies pedestrians to keep a 5 m distance from the drone and that an operator is on the way. This part of the code will come with the second prototype once we receive our initial feedback for prototype one. We have a Tinkercad link that will allow for a clearer test result than that seen in the figures. We have not yet received all our components to be able to build a physical prototype and test it, so for prototype I, we only have the Tinkercad version. As mentioned earlier, the code will be modified to send the automated message, as well as connect it to the other components in the emergency beacon.

Table 6 - Voice System Test Plan for Prototype I

Test ID	Member Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
1	Karen	To test if the wiring of the speaker with the arduino and the test code used give the result expected.	The test method will be done on Tinkercad and will be the execution of the test code.	The speaker made a continuous sound which signifies that both the wiring/setup of the speaker and the initial code function. The code will be used for future prototypes to help modify and add to it to get the expected results. The expected results are that the speaker delivers an automated message to pedestrians.	March 4^{th}, 2021 Test Duration: The test did not last long once the code and the setup was done.	- We can stop when we see reasonable results based on the research done (for prototype I). - We will also stop this stage of testing when we receive feedback.

Figure 8. Arduino Setup for the Voice System

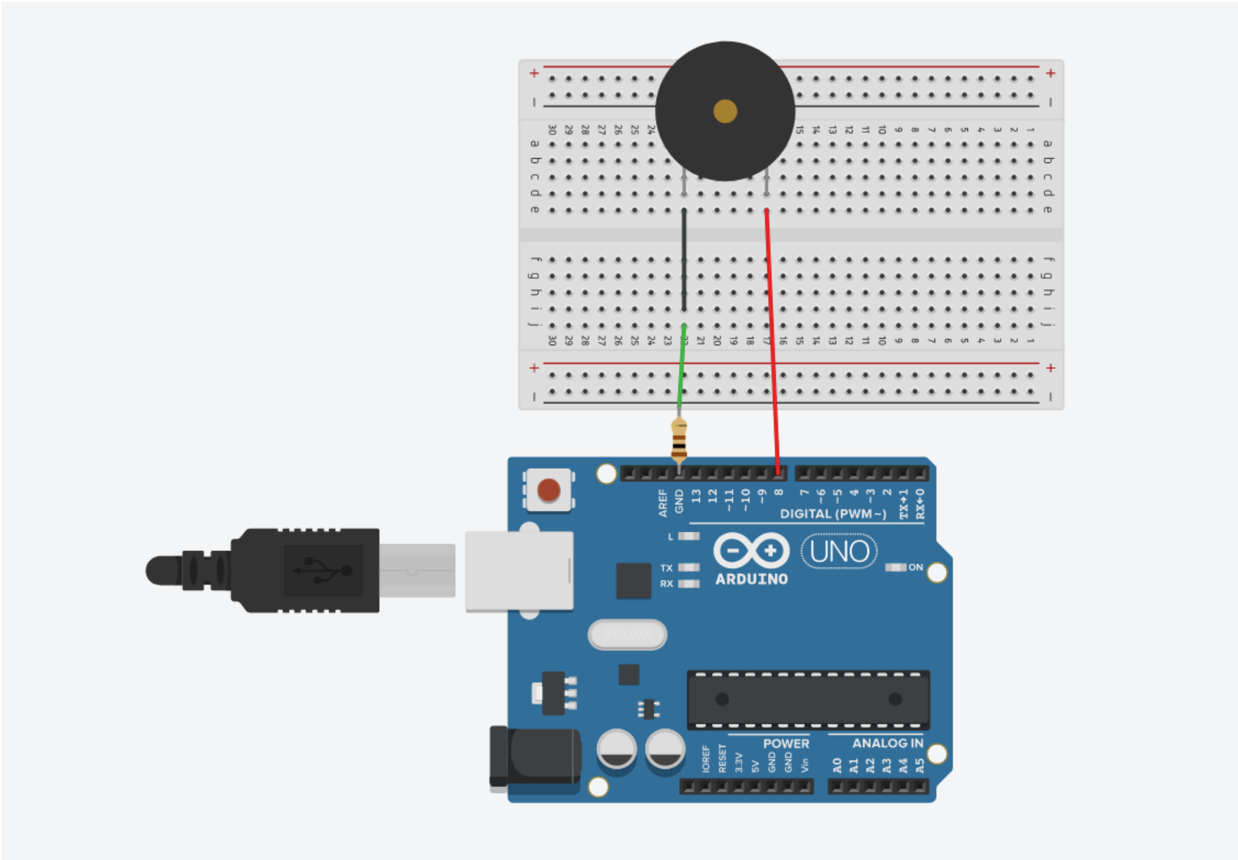
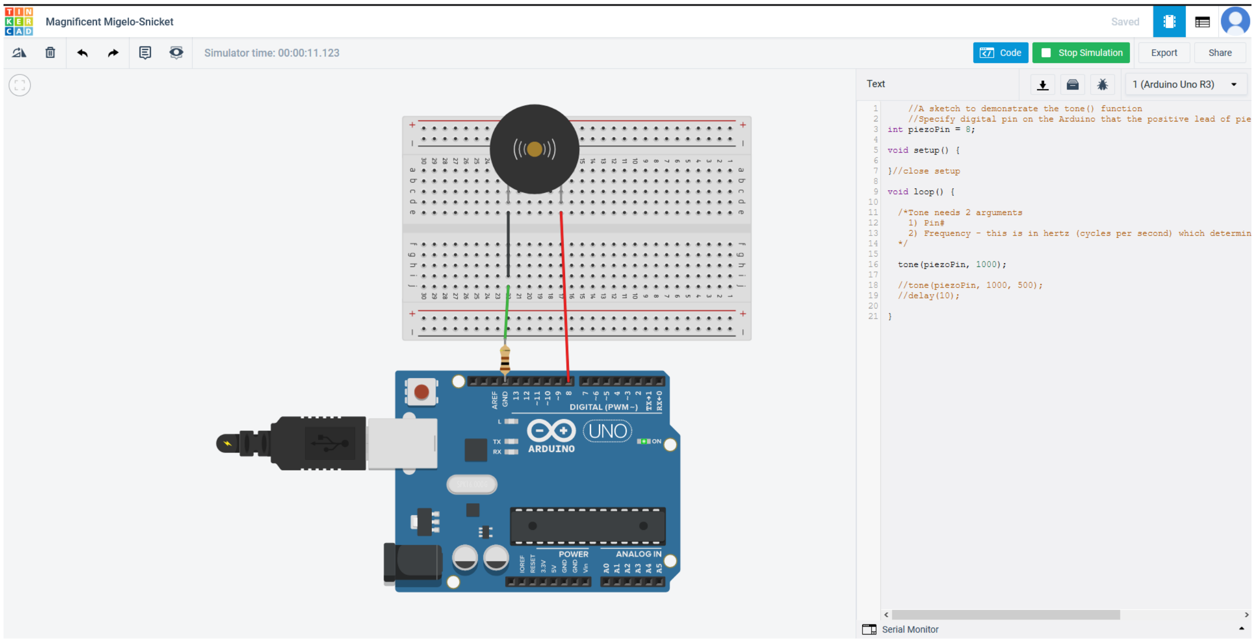


Figure 9. Arduino Code for the Voice System

```
    //A sketch to demonstrate the tone() function
    //Specify digital pin on the Arduino that the positive lead of piezo buzzer is attached.
int piezoPin = 8;

void setup() {

}//close setup

void loop() {

  /*Tone needs 2 arguments
    1) Pin#
    2) Frequency - this is in hertz (cycles per second) which determines the pitch of the noise made
  */

  tone(piezoPin, 1000);

  //tone(piezoPin, 1000, 500);
  //delay(10);

}
```

The code used in this initial testing shown in Figure.., as well as the wiring and setup of the speaker is taken from the following website.

Figure10 . Voice System Test Using Initial Code



A [link to Tinkercad] where I have the setup and the code is also added. This is only in case any further testing is needed.

To complete this part, we used the previously learned knowledge from lab 5 to help guide us in the right direction. In lab 5, we learned how to connect a speaker to a breadboard and then connect it to an Arduino. We also learned how to use a code to test our wiring and what a successful connection looks and sounds like. For the voice system setup, we used a similar setup as that used in lab 5, but we had to make some adjustments as we do not have buttons in our prototype and we are only using one resistor. Based on these adjustments, we had to find a different code to help us test our initial prototype. It was helpful to have something to compare our results with, as it helped us confirm that both our wiring and our initial code was correct.

Light Subsystem

The first light subsystem prototype is shown in the first image below followed by an image of the activated system and the code used to test it. A link to the Tinkercad circuit has also been included for an interactive version of the system. Since we are still waiting to pick up LED lights for the physical prototype, the resistors used in the diagram were based on an estimate and are subject to change at this point. The remainder of the components are the same as parts that we have purchased. The purpose of the code at this point was to ensure that the subsystem was wired correctly and was physically possible. The code will eventually be changed to work in unison with the speakers attached and the timing of the lights may also be modified.

Table 7 - Light System Test Plan for Prototype I

Test ID	Member Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptio-ns of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
1	Jacob	To test the test code and setup of the LED lights in the voice and light subsystem.	The test method will be done on Tinkercad and will be the execution of the test code and subsystem wiring.	The 5 LED lights turned on when the code was activated. This code will be modified to activate after a threshold altitude has been met.	March 4th, 2021 Test Duration: 5 minutes	When the lights are able to turn on and off to ensure the subsystem is wired properly.

Figure 11. Light Subsystem

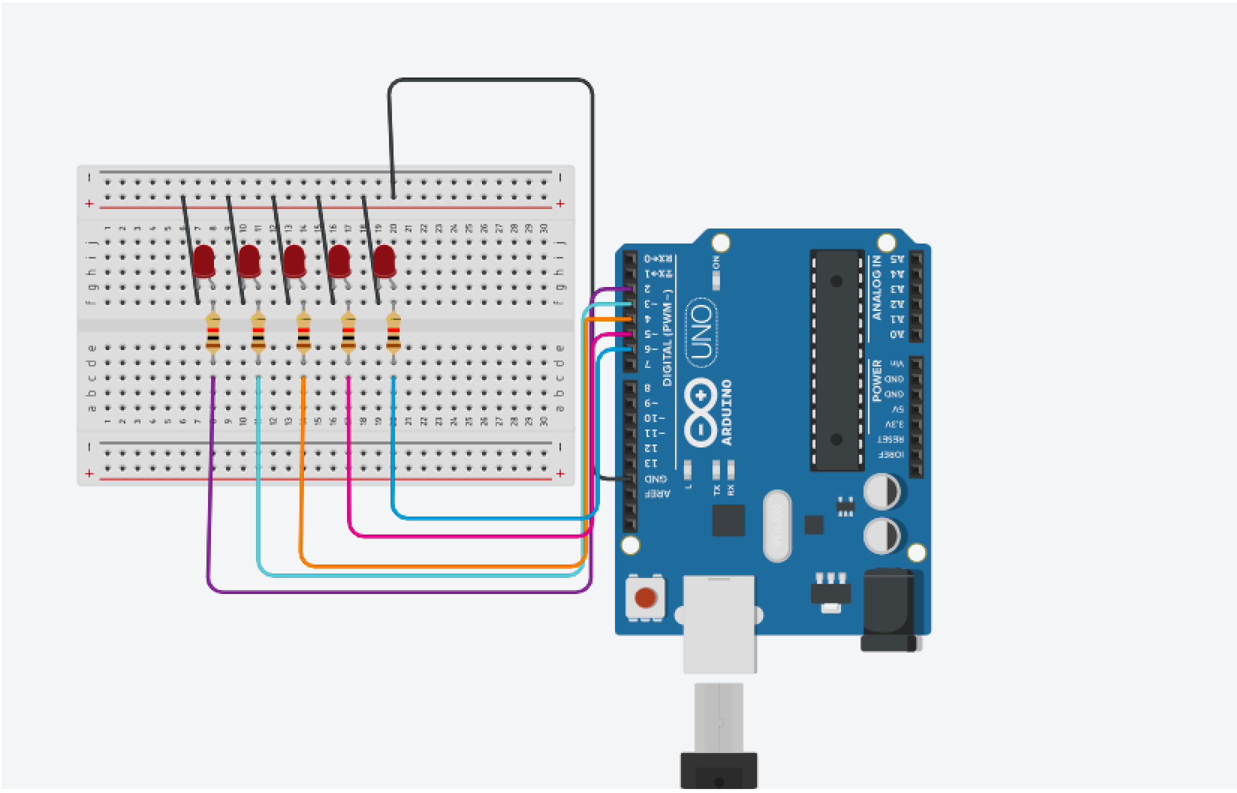


Figure 12. Light Subsystem Test

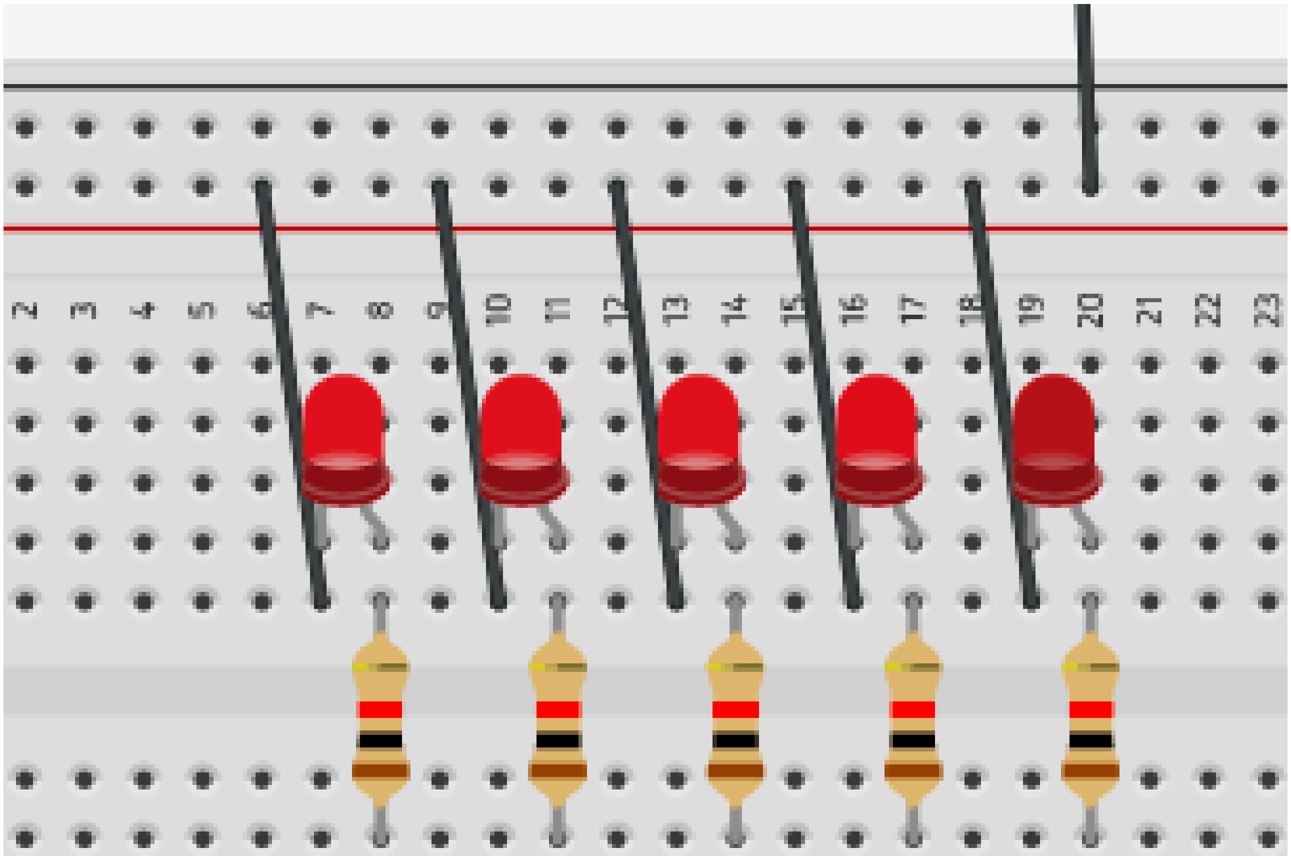


Figure 13. Light Subsystem Test Code

```
int led1 = 2;
int led2 = 3;
int led3 = 4;
int led4 = 5;
int led5 = 6;

int invalue;

void setup ( )
{
  pinMode ( led1, OUTPUT);
  pinMode ( led2, OUTPUT);
  pinMode ( led3, OUTPUT);
  pinMode ( led4, OUTPUT);
  pinMode ( led5, OUTPUT);
}

void loop ( )
{
 digitalWrite (led1, HIGH);
 delay(100);

 digitalWrite (led2, HIGH);
 delay(100);

 digitalWrite (led3, HIGH);
 delay(100);

 digitalWrite (led4, HIGH);
 delay(100);

 digitalWrite (led5, HIGH);
 delay(100);

 digitalWrite(led1, LOW);
 delay(100);

 digitalWrite(led2, LOW);
 delay(100);

 digitalWrite (led3, LOW);
 delay(100);

 digitalWrite (led4, LOW);
 delay(100);

 digitalWrite (led5, LOW);
 delay(100);
}
```

To create and execute this test, I used previous knowledge learned from lab 4 on connecting LED lights and the Arduino.

Interconnections

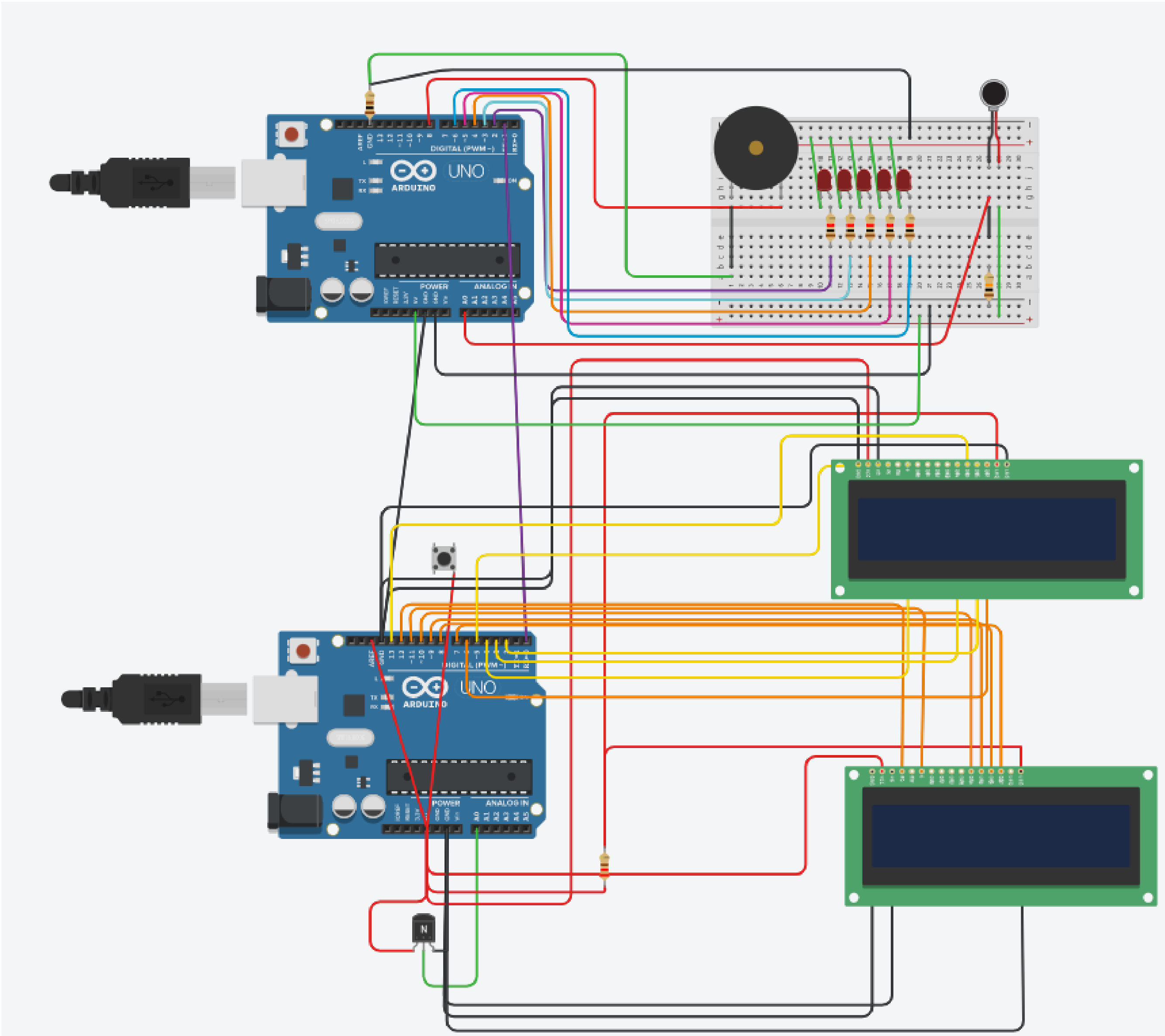
The interconnection for the first prototype includes the wiring of the four subsystems which are the altitude, location, voice and light subsystems. The circuit was visually represented and tested through Tinkercad. The first figure shows the circuit setup with the four above subsystems. Next came the 9 screenshots of the code used for the testing phase. Then are the two figures corresponding to the visual result and the executed result shown in the serial monitor. The second prototype will use the Tinkercad circuit as a guide to build it in person. The code will be then adjusted in order to connect each subsystem to other components. The circuit is expected to be wired properly as well as each component connected is able to run coherently.

Table 8: Interconnections Test plan for the Prototype I

Test ID	Member Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
1	Tri	To test the wiring and execution of the code of the altitude, location, voice and light subsystem.	The test method will be done on Tinkercad consisting of the wiring setup and code test.	As the code was activated, all five LED lights turned on consecutively. The pressure data printed on the Serial Monitor. The speaker, however, would not make continuous sound as when it was tested by Karen, instead, it would make a sound every 1-2 seconds. In addition, the GPS data including the latitude and longitude, as well as the local time was shown up on the LCD screen.	March 4th, 2021 Test Duration: The test only lasted approximately 2 minutes.	- Since the complete prototype runs nearly immediately as the code is executed; the test will stop as the data output of location and altitude is consistent and accurate enough. -The test will also be stopped when receiving feedback.

The knowledge obtained from lab 4 and lab 5 assists the wiring connection of the four above subsystems to Arduino and also the code executed.

Figure 14. The complete wiring connect of the four subsystems



The first figure visually shows how the components are wired together to the Arduino. There consists of the speaker, the vibration motor, two LCD screens, one breadboard, five of 1-kOhm resistors, one 100 Ohm resistor, 10 kOhm resistor, one 220 Ohm resistor, five LED lights, one pushbutton, and one NPN transistor. They are wired together to make a complete prototype that can produce sound through the speakers and have LED lights that can turn on. Also, the location data is presented on the LCD screen.

Figure 15. Overall System Code

```
#include <LiquidCrystal.h>

#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (0<<ADLAR))

float stepADC = 0.0048828125;

LiquidCrystal lcd1(5, 4, 3, 13, 2, 6);
LiquidCrystal lcd2(12, 11, 10, 9, 8, 7);

char sSecventa[119];

//declarate global deoarece in cazul calculatiilor din ecranul 2
//memoria este deja scrisa si ar fi trebuit o alta initializare
//cu mai multe varibile, etc..
static char *aValori[2];
static char *token;
static char *aValori1[9];
static char *token1;
static char *aValori2[15];
static char *token2;

int contor = 0;
bool ecran = 0;
bool stareButon = 0;
bool stareButonAnterioara = 0;
//---------------------

unsigned int citesteADC(unsigned int adc_input)
{
  ADMUX = adc_input | ADC_VREF_TYPE;

  //delay necesar pentru stabilirea ADC pt tens
  //de intrare analogica
  delayMicroseconds(10);
```

Text

```
  //segmentare secventa 2
  token2 = strtok(aValori[1], ",");
  static int increment2 = 0;
  while (token2 != NULL)
  {
    aValori2[increment2++] = token2;

    token2 = strtok(NULL, ",");
  }

  //incheiere segmentare secvente -------------
}

void ecranulUnu()
{
  //afisare viteza
  static float iViteza = 0;
  if (strcmp(aValori1[0], "$GPVTG") == 0)
  {
    iViteza = atof(aValori1[7]);
  }
    //setare cursor: coloana 0, linia 1
  lcd1.setCursor(0, 1);
  lcd1.print("Vit: ");
  lcd1.print(iViteza);
  lcd1.print("km/h");

  //afisare timp
  static long int temp[4];
  static int iSateliti = 0;
  if (strcmp(aValori2[0], "$GPGGA") == 0)
  {
    //stocare sir
    temp[0] = atof(aValori2[1]);
```

Text

```
      //stocare secunde
      temp[3] = temp[0] % 100;
      temp[0] = temp[0] / 100;
      //stocare minute
      temp[2] = temp[0] % 100;
      temp[0] = temp[0] / 100;
      //stocare ore
      temp[1] = temp[0];

      iSateliti = atoi(aValori2[7]);
    }

    //setare cursor: coloana 0, linia 0
    lcd1.setCursor(0, 0);
    lcd1.print("Timp: ");
    lcd1.print(temp[1]);
    lcd1.print(":");
    lcd1.print(temp[2]);
    lcd1.print(":");
    lcd1.print(temp[3]);


    //setare cursor: coloana 0, linia 3
    lcd2.setCursor(0,0);
    lcd2.print("Nr. sat: ");
    lcd2.print(iSateliti);

    //citire termometru de pe pinul analogic "0"
    unsigned int sensorValue = citesteADC(0);
    //calculare temperatura in functie de valorile din datasheetul termometrului
    float fTemperature = (stepADC*sensorValue-0.5)*100;

    //setare cursor: coloana 0, linia 3
    lcd2.setCursor(0, 1);
```

Text

1 (Arduino Uno R3)

```
  lcd2.print("Temp: ");
  lcd2.print(fTemperature, 2);
  lcd2.print(" C");
  //incheiere prelucrare ecran unu--------
}

//afisare/selectare date pe ecranul doi
void ecranulDoi()
{
  //segmentareSecventa();
  //calculare, afisare latitudine
  static float fLatitudine = 0;
  static float fLongitudine = 0;
  static float fAltitudine = 0;
  if (strcmp(aValori2[0], "$GPGGA") == 0)
  {
    fLatitudine = atof(aValori2[2]);
    fLongitudine = atof(aValori2[4]);
    fAltitudine = atof(aValori2[9]) - atof(aValori2[11]);
  }
  //afisare date
  //setare cursor: coloana 0, linia 1
  lcd1.setCursor(0, 1);
  lcd1.print("Lat: ");
  lcd1.print((int)fLatitudine/100+((int)(fLatitudine)%100+(fLatitudine-(int)(fLatitudine)))/60);
  lcd1.print(aValori2[3]);
  //setare cursor: coloana 0, linia 2
  lcd2.setCursor(0, 0);
  lcd2.print("Long: ");
  lcd2.print((int)fLongitudine/100+((int)(fLongitudine)%100+(fLongitudine-(int)(fLongitudine)))/60);
  lcd2.print(aValori2[5]);
  //setare cursor: coloana 0, linia 3
  lcd2.setCursor(0, 1);
  lcd2.print("Alt: ");
```

Text

```
  lcd2.print(fAltitudine);
  lcd2.print("m");
  //incheiere prelucrare ecran doi--------
}

int led1 = 2;
int led2 = 3;
int led3 = 4;
int led4 = 5;
int led5 = 6;
int piezoPin = 8;
int fsrAnalogPin = 0;
int fsrReading;
void setup (void)
{
  pinMode ( led1, OUTPUT);
  pinMode ( led2, OUTPUT);
  pinMode ( led3, OUTPUT);
  pinMode ( led4, OUTPUT);
  pinMode ( led5, OUTPUT);
  Serial.begin(9600);
  lcd1.begin(2,16);
  lcd2.begin(2,16);
}

void loop (void)
{
  {
    digitalWrite (led1, HIGH);
    delay(100);

    digitalWrite (led2, HIGH);
    delay(100);
```

Text

```
      digitalWrite (led3, HIGH);
      delay(100);

      digitalWrite (led4, HIGH);
      delay(100);

      digitalWrite (led5, HIGH);
      delay(100);

      digitalWrite(led1, LOW);
      delay(100);

      digitalWrite(led2, LOW);
      delay(100);

      digitalWrite (led3, LOW);
      delay(100);

      digitalWrite (led4, LOW);
      delay(100);

      digitalWrite (led5, LOW);
      delay(100);
      tone(piezoPin, 1000, 500);
      }
    {
       fsrReading = analogRead(fsrAnalogPin);
      Serial.print("Analog reading = ");
       Serial.println(fsrReading);

      if (fsrReading < 10) {
        Serial.println(" - No pressure");
       } else if (fsrReading < 200) {
         Serial.println(" - Light touch");
```

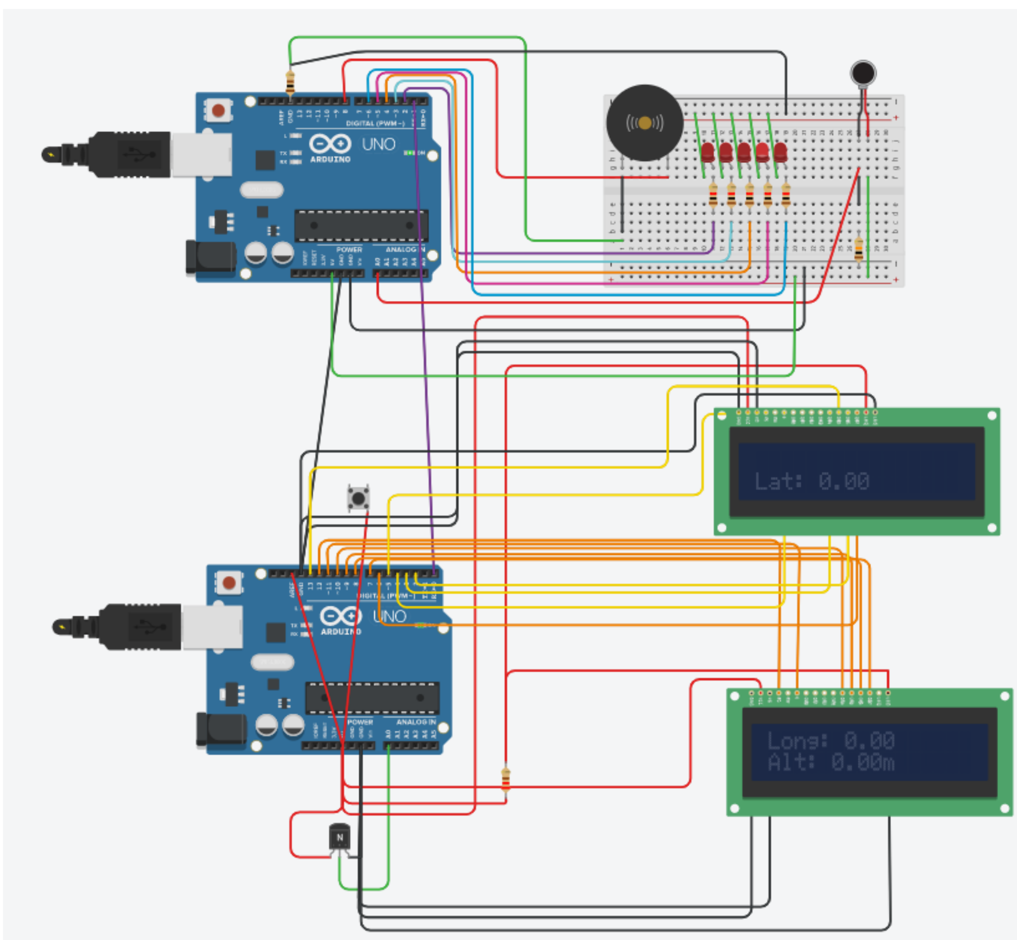
Text

```
     } else if (fsrReading < 500) {
       Serial.println(" - Light squeeze");
     } else if (fsrReading < 800) {
       Serial.println(" - Medium squeeze");
     } else {
      Serial.println(" - Big squeeze");
    }
     delay(1000);
  }
  {
  if(contor < 1)
  {
    Serial.readBytes(sSecventa, 119);
    Serial.println(sSecventa);
    segmentareSecventa();
    contor++;
  }

  //citire stare buton
  /*stareButon = (PIND & (1 << PIND2));

  if(stareButon != stareButonAnterioara)
  {
    delay(200);
    ecran = !ecran;

    if( stareButon == 0)
    {
      stareButon = !stareButon;
    }
  }
  stareButonAnterioara = stareButon;

*/
```

```

    ecranulUnu();
    delay(1000);
    lcd1.clear();
    lcd2.clear();

    ecranulDoi();
    delay(1000);
    lcd1.clear();
    lcd2.clear();
}
}
```

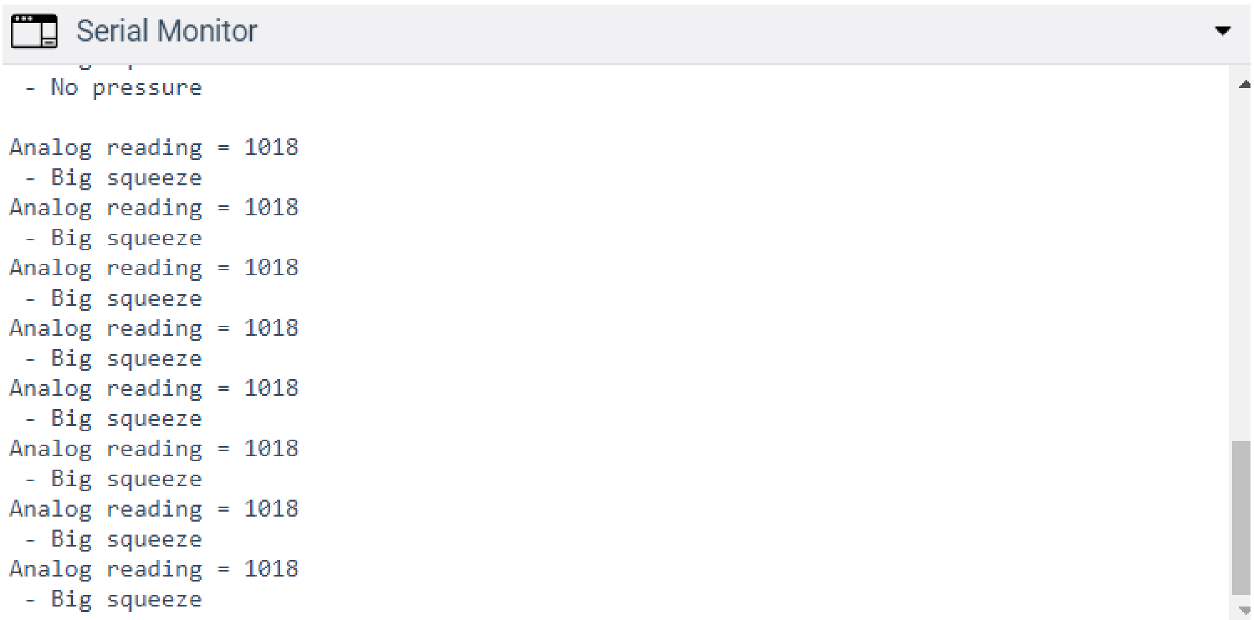
The eight figures above are the code used in the interconnections prototype.

Figure 16. Overall system test visual



As the simulation is started, each of the five LED lights will be lighting on consecutively from left to right and will stay on until the simulation is stopped. Simultaneously, the speaker will make a loud sound continually (the sounds last 1-2 seconds before turning off). On the two LCD screens, all the data about the flight time, latitude, longitude and altitude of the system will be shown.

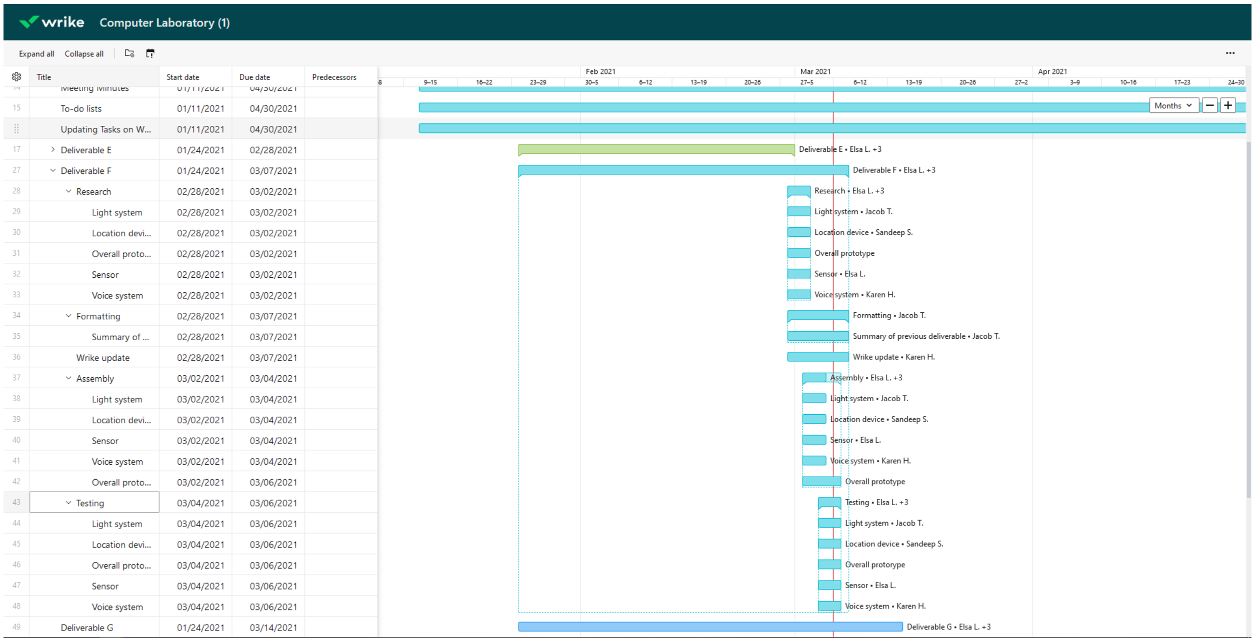
Figure 17. The results shown on the Serial Monitor



For the last figure, the Serial Monitor will print out the data of the pressure after executing the code; the data will be taken as the live results shown on the monitor are consistent.

The link to this circuit and its code can be found here.

Wrike Update



	Title	Start date	Due date	Predecessors
14	Meeting Minutes	01/11/2021	04/30/2021	
15	To-do lists	01/11/2021	04/30/2021	
	Updating Tasks on W...	01/11/2021	04/30/2021	
17	Deliverable E	01/24/2021	02/28/2021	
27	Deliverable F	01/24/2021	03/07/2021	
28	Research	02/28/2021	03/02/2021	
29	Light system	02/28/2021	03/02/2021	
30	Location devi...	02/28/2021	03/02/2021	
31	Overall proto...	02/28/2021	03/02/2021	
32	Sensor	02/28/2021	03/02/2021	
33	Voice system	02/28/2021	03/02/2021	
34	Formatting	02/28/2021	03/07/2021	
35	Summary of ...	02/28/2021	03/07/2021	
36	Wrike update	02/28/2021	03/07/2021	
37	Assembly	03/02/2021	03/04/2021	
38	Light system	03/02/2021	03/04/2021	
39	Location devi...	03/02/2021	03/04/2021	
40	Sensor	03/02/2021	03/04/2021	
41	Voice system	03/02/2021	03/04/2021	
42	Overall proto...	03/02/2021	03/06/2021	
43	Testing	03/04/2021	03/06/2021	
44	Light system	03/04/2021	03/06/2021	
45	Location devi...	03/04/2021	03/06/2021	
46	Overall proto...	03/04/2021	03/06/2021	
47	Sensor	03/04/2021	03/06/2021	
48	Voice system	03/04/2021	03/06/2021	
49	Deliverable G	01/24/2021	03/14/2021	