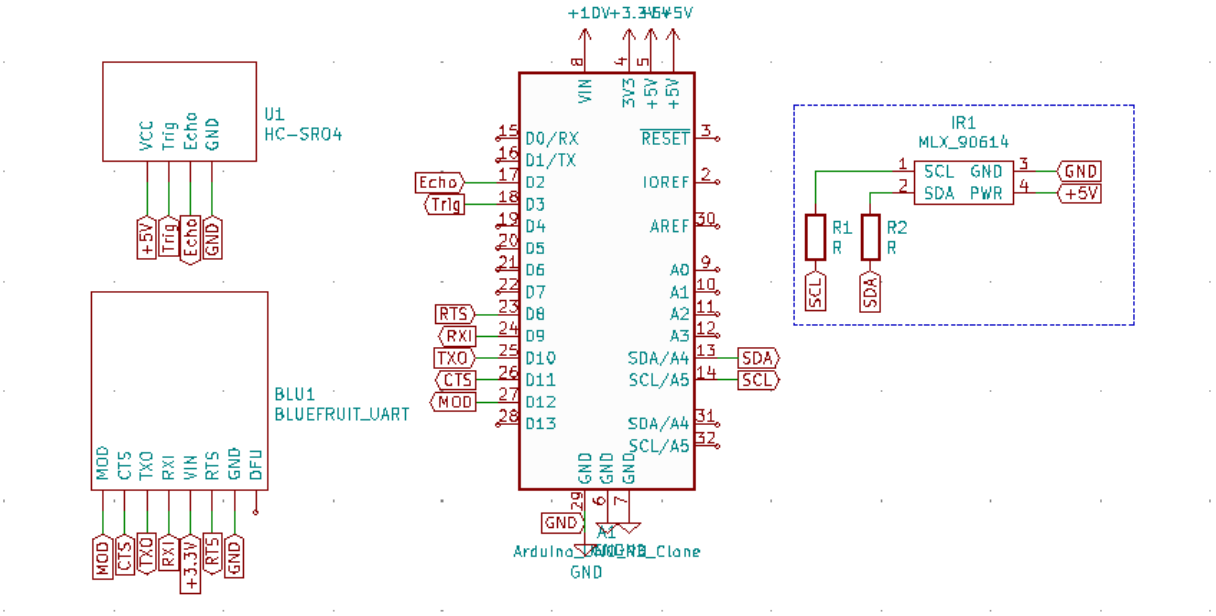
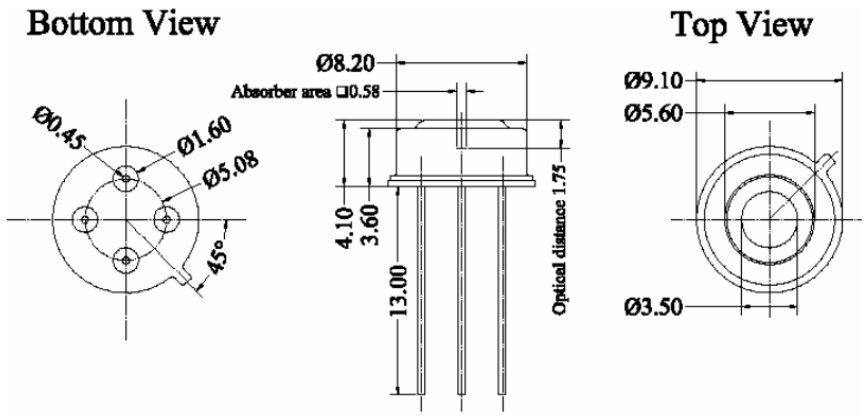
Electrical Schematic

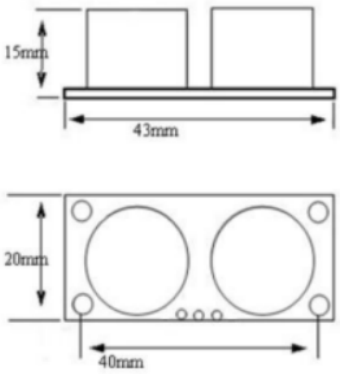


Mechanical Drawings

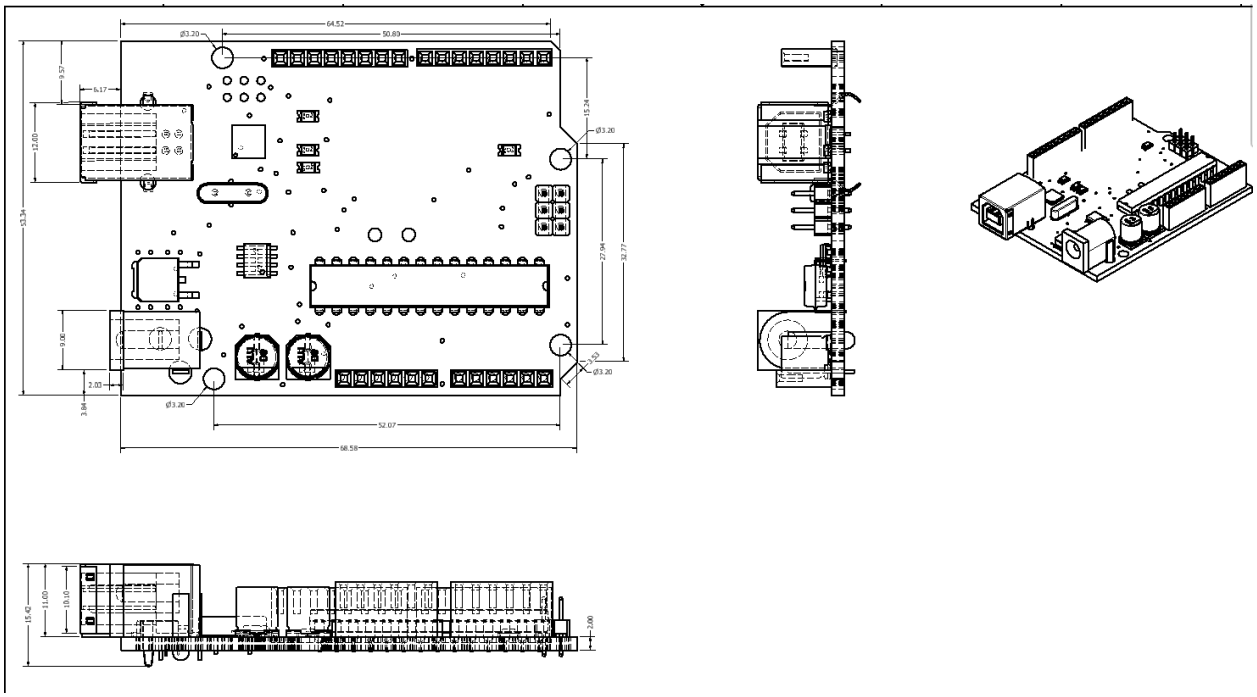
MLX90614:



HC-SR04:

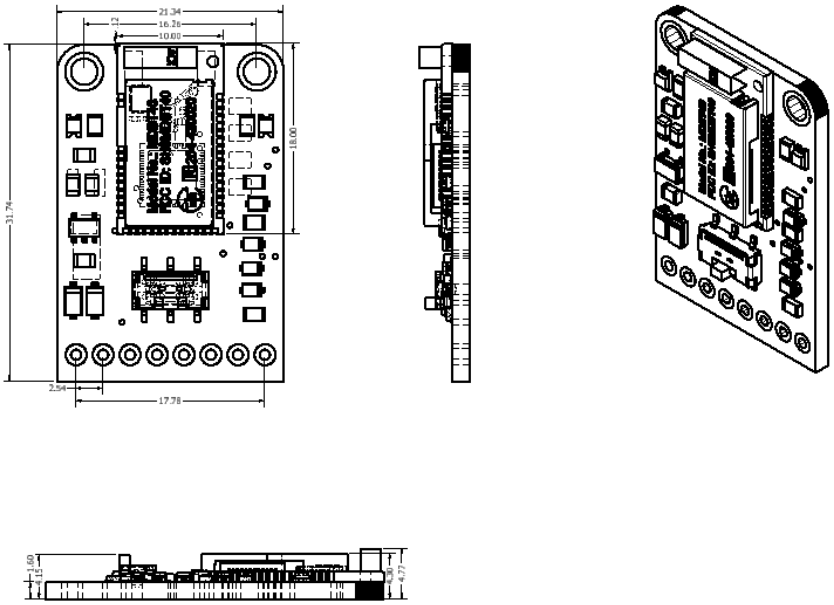


Arduino Uno R3:



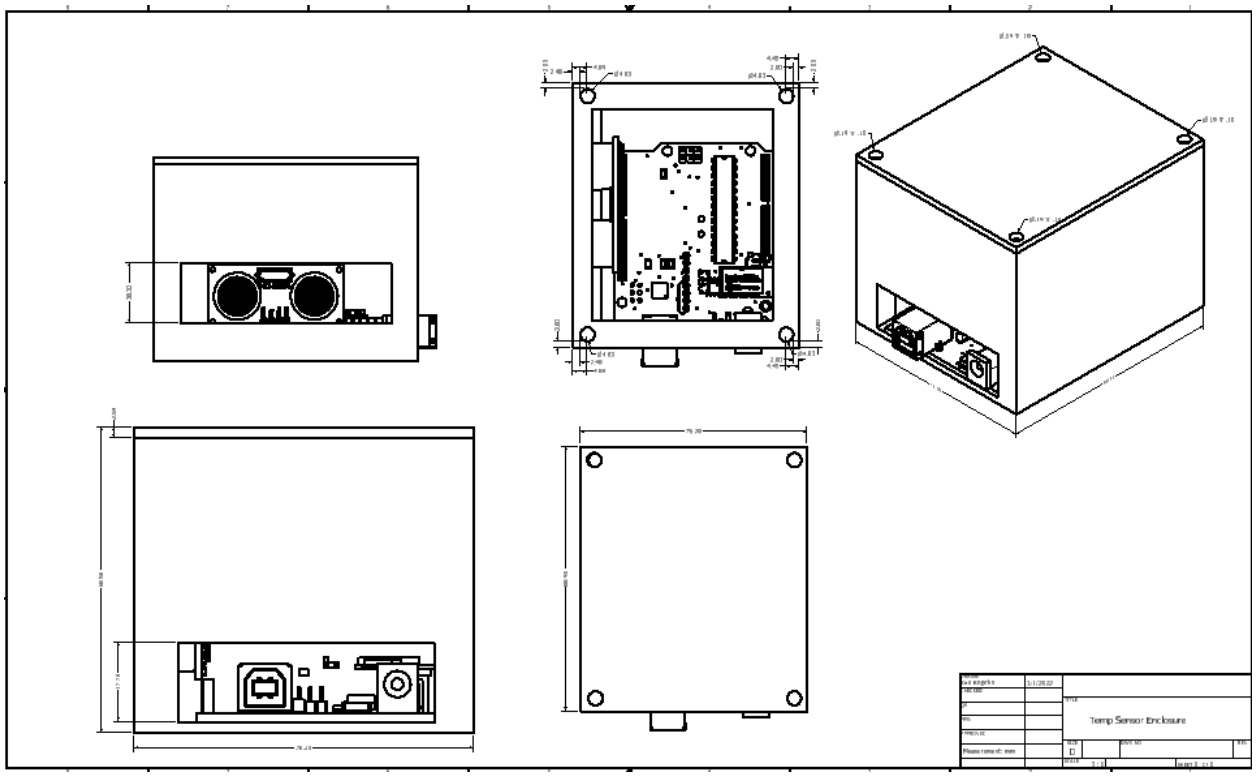
[Full mechanical drawing]
[Link to CAD Model]

Bluefruit BLE UART:



[Full mechanical drawing]
[Link to CAD Model]

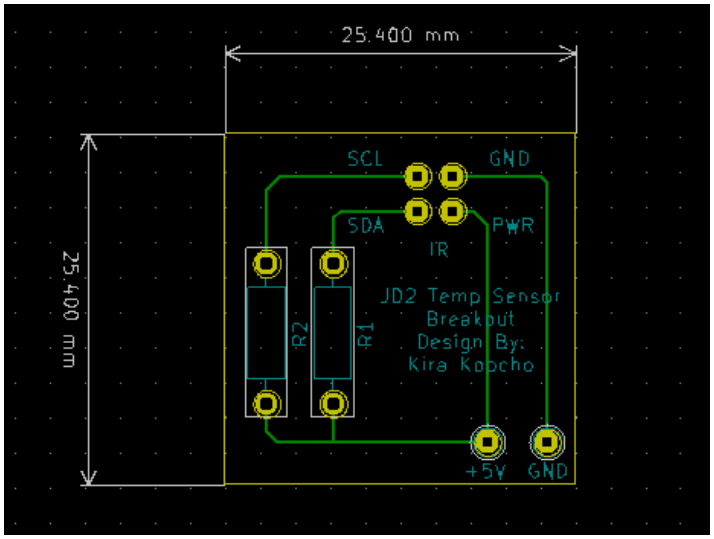
Enclosure:



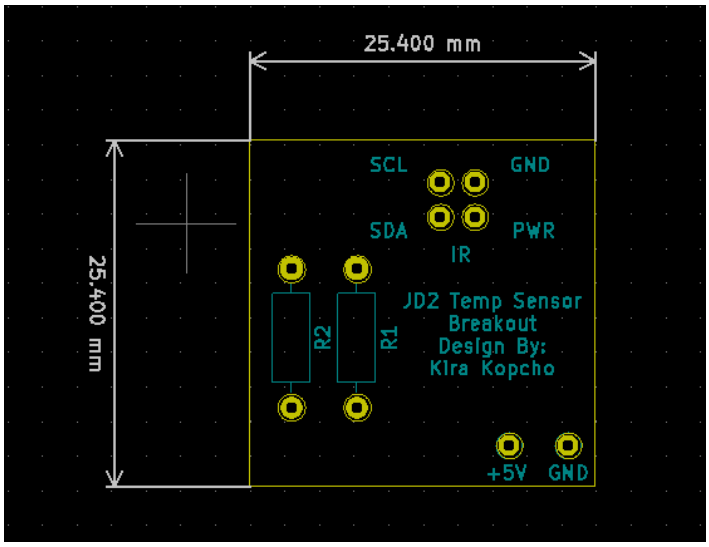
[Full mechanical drawing]

PCB Layers:

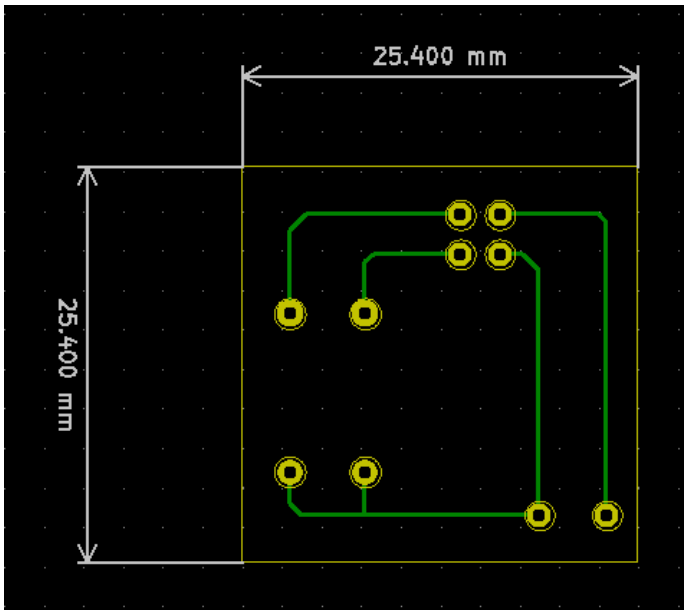
Full PCB:



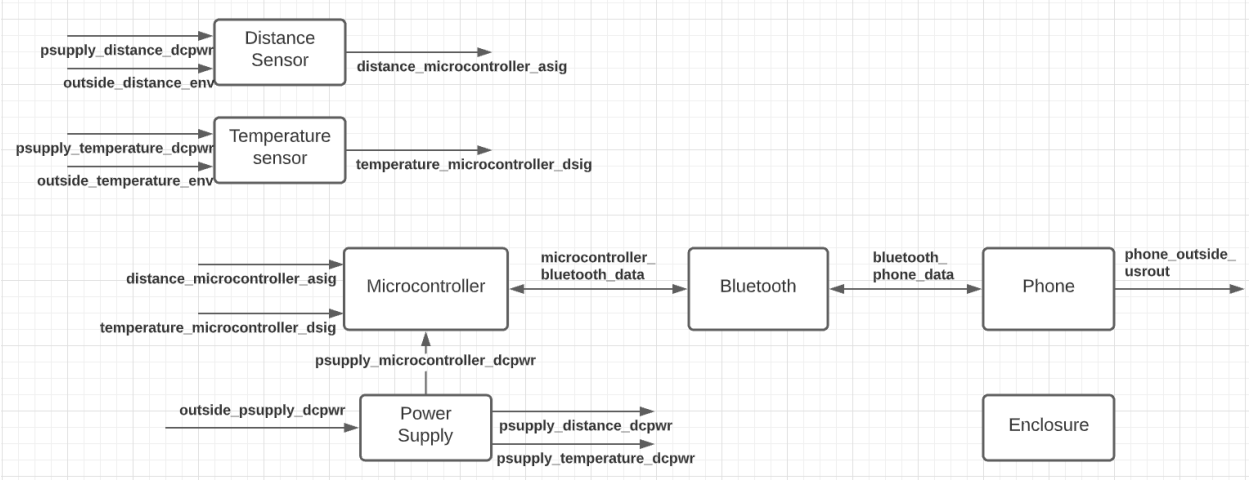
Front Layer:



Back Layer:



Top-Level Block Diagram



Interface Table

Interface Name	Properties
outside_psupply_dcpwr	• V_{max} = 5V • V_{min} = 4.75V • I_{max} = 500mA • I_{min} = 100mA
outside_distance_env	• Ultrasonic sensor will detect how far away the user is
outside_temperature_env	• Infrared sensor will detect the temperature of the object in front of it
psupply_distance_dcpwr	• V_{max} = 5.5V • V_{min} = 4.5V • I_{max} = 10mA • I_{min} = 20mA
psupply_temperature_dcpwr	• V_{max} = 5.5V • V_{min} = 4.5V • I_{max} = 2mA • I_{min} = 1.3mA
psupply_microcontroller_dcpwr	• V_{max} = 5.5V • V_{min} = 2.7V • I_{max} = 500mA • I_{min} = 100mA
distance_microcontroller_asig	• V_{max} = 5V • V_{min} = 3V • Distance is proportional to the time when signal is high • Distance = velocity / time

temperature_microcontroller_dsig	• $Frequency_{clk\_max}$ = 100kHz • $Frequency_{clk\_min}$ = 10kHz • V_{clk_max} = 5.5V • V_{clk_min} = 4.5V • I_{clk_max} = 2mA • I_{clk_min} = 1.3mA
microcontroller_bluetooth_data	• Communicates with Microcontroller using SPI protocol • V_{max} = 5V • Uses a baud rate of 9600 • Uses UART protocol
bluetooth_phone_data	• Low Energy Bluetooth allows arduino to send data to phone using adapter • App receives data transmitted from bluetooth transmitter
phone_outside_usrout	• Phone acts as a display for the temperature sensor system • User can check their temperature and the previous scans on the phone

Bill of Materials

Reference Designator	**Part #**	**Description**	**Quantity**	**Cost**
IR1	MLX90614-AAA	Temperature Sensor	1	$15.95
U1	HC-SR04	Ultrasonic Sensor	1	$3.95
BLU1	MDBT40-P256R *	Bluefruit Bluetooth UART Module	1	$17.50
A1	Uno R3	Arduino Uno Clone	1	$6.00

***** This is the part number for the specific bluetooth antenna used on the Bluefruit Module: the board itself has no part number

Code

Arduino code:

```
/**
* Temperature Sensor Main Code
* Bluetooth Module by: Patrick Liang
```

```
* Ultrasonic and Temperature Sensor Modules by: Kelton Luu
* Main Arduino Code by: Kira Kopcho
*/

#include <Arduino.h>
#include <Adafruit_MLX90614.h>
#include <SPI.h>
#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"

#include "BluefruitConfig.h"

#if SOFTWARE_SERIAL_AVAILABLE
 #include <SoftwareSerial.h>
#endif

/*=========================================================================
   APPLICATION SETTINGS

   FACTORYRESET_ENABLE       Perform a factory reset when running this sketch
                             Enabling this will put your Bluefruit LE module
                             in a 'known good' state and clear any config
                             data set in previous sketches or projects, so
                             running this at least once is a good idea.
                             When deploying your project, however, you will
                             want to disable factory reset by setting this
                             value to 0. If you are making changes to your
                             Bluefruit LE device via AT commands, and those
                             changes aren't persisting across resets, this
                             is the reason why.  Factory reset will erase
                             the non-volatile memory where config data is
                             stored, setting it back to factory default
                             values.
                              Some sketches that require you to bond to a
                             central device (HID mouse, keyboard, etc.)
                             won't work at all with this feature enabled
                             since the factory reset will clear all of the
                             bonding data stored on the chip, meaning the
                             central device won't be able to reconnect.
   MINIMUM_FIRMWARE_VERSION  Minimum firmware version to have some new features
   MODE_LED_BEHAVIOUR        LED activity, valid options are
                             "DISABLE" or "MODE" or "BLEUART" or
                             "HWUART"  or "SPI"  or "MANUAL"
   -------------------------------------------------------------------------*/
   #define FACTORYRESET_ENABLE         0
   #define MINIMUM_FIRMWARE_VERSION    "0.6.6"
   #define MODE_LED_BEHAVIOUR          "MODE"
/*=========================================================================*/
```

```
// Create the bluefruit object, either software serial...uncomment these lines

SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_TXD_PIN,
BLUEFRUIT_SWUART_RXD_PIN);

Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,
                      BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);


// A small helper
void error(const __FlashStringHelper*err) {
 Serial.println(err);
 while (1);
}

Adafruit_MLX90614 mlx = Adafruit_MLX90614(); //creates an object for the mlx
sensor

//begin pin defines
int gndPin = 7; // gnd pin for distance sensor
int echoPin = 6; // attach pin D2 Arduino to pin Echo of HC-SR04
int trigPin = 5; //attach pin D3 Arduino to pin Trig of HC-SR04
int vccPin = 4; //Vcc pin for distance sensor

//SDA: A4
//SCL: A5

//global variables for distance sensor
float distance = 0; //distance to the object
float tooCloseVal = 1; //when the object is too close to the sensor
float maximum = 2000; //when object is too close that sensor does not work
appropriately
float tooFarVal = 10;
long duration; //how long the sound wave travels before detecting something

//global variables for temp sensor
double temperature = 0;

void setup() {
 //begin I/O defines
 pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
 pinMode(vccPin, OUTPUT); // Sets the vccPin as an OUTPUT
 pinMode(gndPin, OUTPUT); // Sets the gndPin as an OUTPUT
 pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT

 digitalWrite(vccPin, HIGH);
 digitalWrite(gndPin, LOW);
```

```
 //Begin Setup for Temperature Sensor
 while (!Serial);
 Serial.begin(115200);
 Serial.println("Begin Setup");

 if (!mlx.begin()) {
   Serial.println("Error connecting to MLX sensor. Check wiring.");
   delay(100);
 };

 Serial.println(F("Adafruit Bluefruit Command <-> Data Mode Example"));
 Serial.println(F("------------------------------------------------"));

 /* Initialise the module */
 Serial.print(F("Initialising the Bluefruit LE module: "));

 if ( !ble.begin(VERBOSE_MODE) )
 {
   error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode & check
wiring?"));
 }
 Serial.println( F("OK!") );

 if ( FACTORYRESET_ENABLE )
 {
   /* Perform a factory reset to make sure everything is in a known state */
   Serial.println(F("Performing a factory reset: "));
   if ( ! ble.factoryReset() ){
     error(F("Couldn't factory reset"));
   }
 }
//
 /* Disable command echo from Bluefruit */
 ble.echo(false);

 Serial.println("Requesting Bluefruit info:");
 /* Print Bluefruit information */
 ble.info();

 Serial.println(F("Please use Adafruit Bluefruit LE app to connect in UART
mode"));
 Serial.println(F("Then Enter characters to send to Bluefruit"));
 Serial.println();

 ble.verbose(false);  // debug info is a little annoying after this point!
//
 /* Wait for connection */
 while (! ble.isConnected()) {
     delay(500);
```

```
 }

 Serial.println(F("******************************"));

 // LED Activity command is only supported from 0.6.6
 if ( ble.isVersionAtLeast(MINIMUM_FIRMWARE_VERSION) )
 {
   // Change Mode LED Activity
   Serial.println(F("Change LED activity to " MODE_LED_BEHAVIOUR));
   ble.sendCommandCheckOK("AT+HWModeLED=" MODE_LED_BEHAVIOUR);
 }

 // Set module to DATA mode
 Serial.println( F("Switching to DATA mode!") );
 ble.setMode(BLUEFRUIT_MODE_DATA);

 Serial.println(F("******************************"));

}

void sonarDistance(){
 digitalWrite(trigPin, LOW); //clears data on the sensor
 delayMicroseconds(5); //5us delay
 digitalWrite(trigPin, HIGH); //starts the sensor if Microseconds > 10
 delayMicroseconds(15); //15 us delay
 digitalWrite(trigPin, LOW); //stops sensor
 duration = pulseIn(echoPin, HIGH); //gives the distance from the sensor and
object
 distance = duration*0.0343/2; //calculate the distance in centimeters, d = v *
t
//  Serial.print("Distance "); //print message to serial montior
//  Serial.print(distance); //print the calculated distance to serial monitor
//  Serial.println(" cm"); //print message to serial monitor
}

void loop() {
 //All inputs below are hardcoded for testing purposes!

 bool tooClose = false;
 bool tooFar = false;
 bool scan = false; //(Scan is triggered by app)
 //char n, inputs[BUFSIZE+1];

 sonarDistance(); //call function to calculate the distance
 //delay(2000); //delay for 2 seconds

/**
* Code for gathering if scan has been initiated*/
 // Echo received data
```

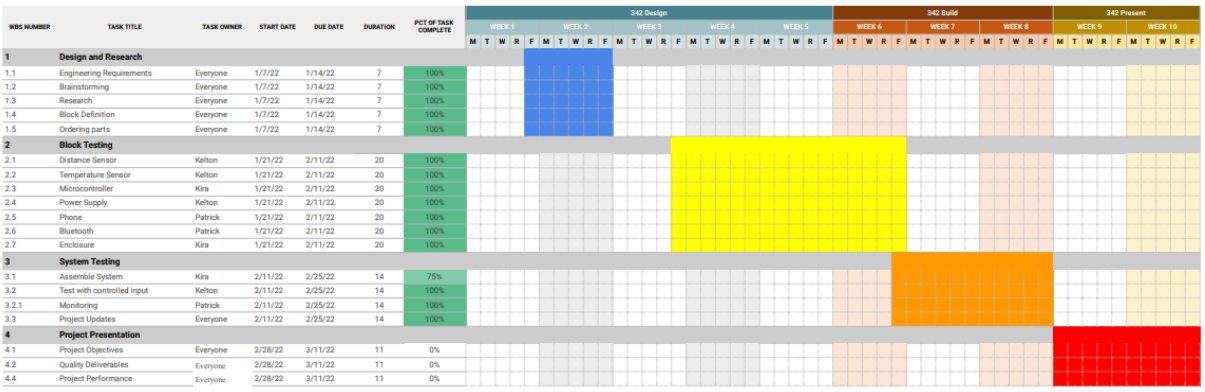
```
 String string = "";
 while ( ble.available() )
 {
   int c = ble.read();
   string += (char)c;
 }
 if(string.equals("get")){
   scan = true;
 }

if(scan){
 //Temperature is only displayed if the user is within range
 if(distance < tooCloseVal || distance > maximum){
   tooClose = true;
   ble.print("too close");
 }
 else if(distance > tooFarVal){
   tooFar = true;
   //Serial.println("Too Far, Please move forward");
   ble.print("too far");
 }
 else{
   temperature = mlx.readObjectTempF();
   ble.print((String)temperature); /* -> casts temp float to string */
 }
}

}
```

Phone App Code available at the github link listed below:
https://github.com/pl215364731/Temperature_sensor_app.git

Time Report:



WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**Design and Research**					
1.1	Engineering Requirements	Everyone	1/7/22	1/14/22	7	100%
1.2	Brainstorming	Everyone	1/7/22	1/14/22	7	100%
1.3	Research	Everyone	1/7/22	1/14/22	7	100%
1.4	Block Definition	Everyone	1/7/22	1/14/22	7	100%
1.5	Ordering parts	Everyone	1/7/22	1/14/22	7	100%
2	**Block Testing**					
2.1	Distance Sensor	Kelton	1/21/22	2/11/22	20	100%
2.2	Temperature Sensor	Kelton	1/21/22	2/11/22	20	100%
2.3	Microcontroller	Kira	1/21/22	2/11/22	20	100%
2.4	Power Supply	Kelton	1/21/22	2/11/22	20	100%
2.5	Phone	Patrick	1/21/22	2/11/22	20	100%
2.6	Bluetooth	Patrick	1/21/22	2/11/22	20	100%
2.7	Enclosure	Kira	1/21/22	2/11/22	20	100%
3	**System Testing**					
3.1	Assemble System	Kira	2/11/22	2/25/22	14	75%
3.2	Test with controlled input	Kelton	2/11/22	2/25/22	14	100%
3.2.1	Monitoring	Patrick	2/11/22	2/25/22	14	100%
3.3	Project Updates	Everyone	2/11/22	2/25/22	14	100%
4	**Project Presentation**					
4.1	Project Objectives	Everyone	2/28/22	3/11/22	11	0%
4.2	Quality Deliverables	Everyone	2/28/22	3/11/22	11	0%
4.4	Project Performance	Everyone	2/28/22	3/11/22	11	0%

Chart also available at:
https://docs.google.com/spreadsheets/d/14lurAnqDdQPQSKtfm9YJVZyqJ4kYkQFJulCPrAbgrh0/edit?usp=sharing