ARDUINO IOT WEATHER STATION

SUMMARY

The Arduino Weather System comes packed with features. It can measure the temperature, humidity, luminous intensity and even dust particulate density. Furthermore, using a Bluetooth module, it allows the microcontroller to transmit information to a smartphone. It is portable and uses little power, which allows it to run for a few months before recharging.

RESEARCH

There are quite a few components that I will have to choose for this project, such as the sensors and the microcontroller. This weather station should last as long as possible on a single charge, hence the sensors that I used must be power efficient and compatible with the microcontroller platform.

OPTIMISING FOR POWER

Through my research, I found out 3 different microcontrollers, which uses different programming languages. Of the 3, I decided to go with Arduino Pro Mini, which uses less power (100 times less). This is because Arduino Pro Mini does not come with a programmer, reducing the power consumption, which will be suitable for my battery-powered project. However, I will have to use a separate programmer to upload the codes. To further improve the power efficiency, I made some modifications to the microcontroller, to make it more efficient.

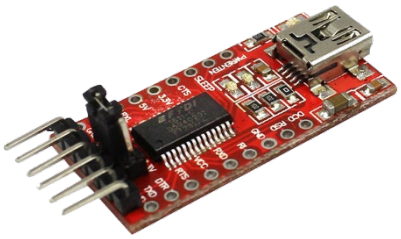


Figure 1: FTDI Programmer

CONSTRUCTION OF ELECTRONICS

PLANNING

I started by using Fritzing to design an electronic circuit. While creating the circuit, I also planned out how the cables should be wired, so that the Arduino Microcontroller can read and write data from the different modules. The modules and their purposes will be described in the later part. I also designed a screen layout for the LCD display, which I will follow when programming the microcontroller. (The microcontroller will be changed to Arduino Pro Mini)

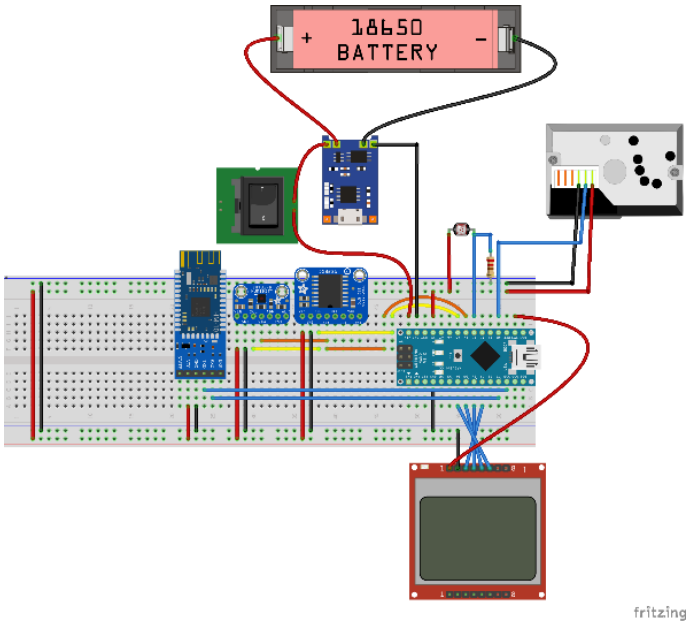


Figure 2: Fritzing Circuit

OPTIMISING FOR POWER EFFICIENCY

For the Software side, I changed the driver of the LCD which resulted in a 75% reduction of power usage. Furthermore, I powered off the sensors when they are not in used, hence minimising the amount of energy used. I also set the microcontroller to low power mode between measurements. Therefore, the microcontroller will only collect data from the sensor once every 8 seconds, instead of continuously, which will reduce its power usage.

For the circuit, I removed the power LED (yellow circle) from the Arduino Pro Mini. As this LED is always powered to indicate that the Arduino is on, it does not have any meaningful application for the project as the Arduino Microcontroller will be hidden inside an enclosure.

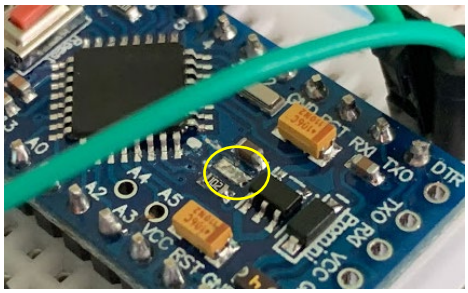


Figure 3: Removed Power LED Indicator

EXECUTION

STEP 1: CYTRON ITEMS ARRIVED

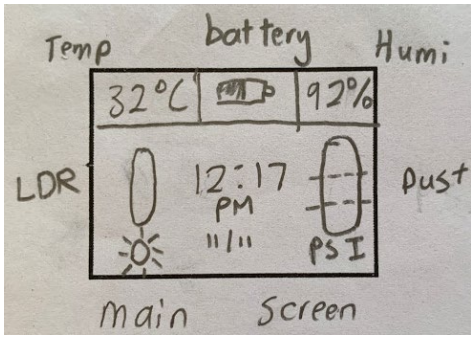
I received the items I ordered from Cytron within 2 days. As most of the items I purchased from Cytron are meant for mounting and circuit board, I could not start programming. However, I did manage to create a light intensity circuit with the items available. Furthermore, it is also during this time where I start planning the screen layout for the LCD Display.

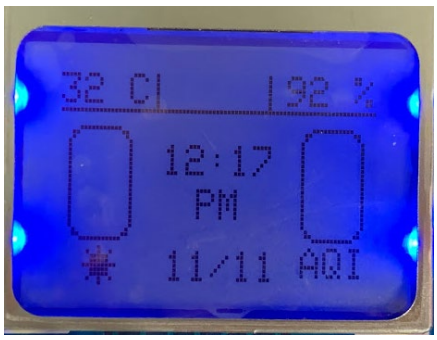
Using the Light intensity sensor, I managed to create a simple program which read and output the LDR sensor value to my laptop.

Figure 4: Cytron Delivery Items

STEP 2: LCD DISPLAY

The LCD Display I ordered arrived about 1 week after I ordered. Using the Drawing of that layout that I created for the LCD I began to create the code which follows the layout of my drawing. As most sensors for this project have not arrived, I used “fake” data as a placeholder. These values will be replaced by actual sensor values as they arrive.

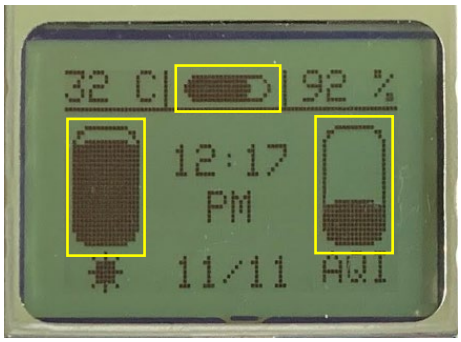


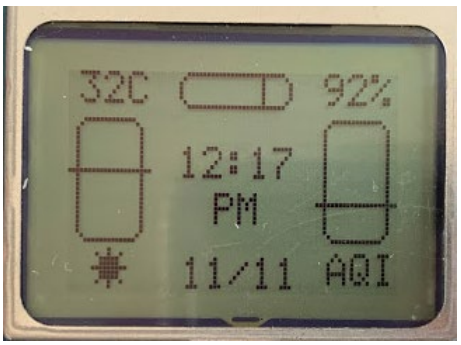


STEP 3: REDUCING POWER CONSUMPTION OF LCD DISPLAY

When I measured the LCD Current draw, I made a worrying discovery. With the LCD Backlight off, it is drawing 4mA of current. The power consumption is about 20 times more than what I would expect. According to the datasheet, it should only draw 0.2mA of current.

After some research, I realised that the driver which I used, is not optimised. Hence, it causes the LCD to draw more current than usual. To solve this, I tried a different driver. This solves the high-power consumption issue but created another issue. With the alternative driver, some of the display functions are not supported, such as drawing a filled shape (which are used in the battery, light and AQI [Air Quality Index] measurements). To solve this issue, I created an alternative method, which uses lines as an indicator, to display the data without using the fill function.





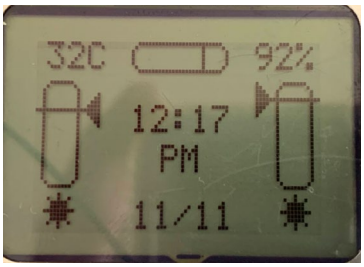
[UNSUCESSFUL] AIR QUALITY SENSOR

I could not get a good Air Quality reading using the sensor that I purchased. This could be due to interference, inaccurate sensor, defective product, wrong implementation, etc. I have tested it with different drivers, example codes and microcontrollers. However, all the outputs from the sensor are inaccurate, ranging from 0% dust density to 100% dust density – the maximum measurable density. This got me to conclude that the sensor outputs false readings, when compared to NEA's Air Quality Readings. I decided to not implement the AQI sensor in the project and replace it with another light intensity sensor instead. The sensors will be installed at both the left and the right side of the enclosure, which provides the user with a better description of their surrounding brightness level.

STEP 3.5: LIGHT INTENSITY SENSORS

The code will take 10 measurements and take the mean value, which provides for a more accurate reading.

Furthermore, I also refined the brightness bars, which in my opinion, is a better-looking layout. The left bar represents the left brightness level and right bar represents the right brightness level. The light brightness bar works by calculating the percentage between the brightest value the code recorded and the current brightness value. For example, if the brightest value is 500lux, and the sensor measures 250lux, the display bar will be at 50%. If the measured value is higher than the maximum brightness value stored, the code will update the maximum brightness value to be the measured value.




```
for (int x = 0; x < 10; x++) {
  sensorValueL += analogRead(ldrL);
  sensorValueR += analogRead(ldrR);
}
sensorValueL = sensorValueL / 10;
sensorValueR = sensorValueR / 10;
digitalWrite(lightEnable, LOW);

if (sensorValueL > maxLight) {
  maxLight = sensorValueL;
}
if (sensorValueR > maxLight) {
  maxLight = sensorValueR;
}
```

STEP 4: BLUETOOTH

Sometimes, I would like to get a more accurate value than the values shown on the display. This can be achieved with a Bluetooth Module. Using Bluetooth, I can obtain the exact sensor outputs and even error messages, which cannot be displayed on the small display. The Bluetooth module provided me with a unique learning experience, as I can learn does an Arduino communicate to the Bluetooth module, which communicates with my mobile phone and displays the data.

The details on the display is updated every 8 seconds. However, when I connect my device to the Bluetooth module, the Arduino microcontroller will detect it and increase the update interval to 1 second. This will allow the user to see the most up to date information, which they will most likely want, when connecting their Arduino to Bluetooth.

[put final image]

STEP 5: TEMPERATURE AND HUMIDITY SENSOR

The temperature and humidity sensor – SHT35 communicate with the Arduino using I2C protocol, which provides a simple interface to measure the temperature and humidity values. However, the pins are not pre-soldered.

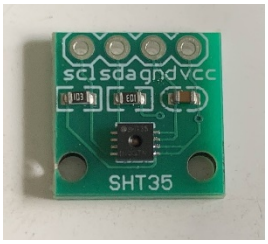
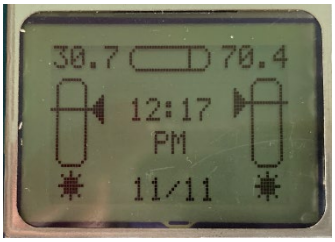


Figure 5: Temperature and Humidity Sensor

After soldering the pins, I proceed to test the sensor, which provides a 2dp accuracy on both humidity and temperature. While I exhale on the sensor, I can see the temperature and humidity level rise. To further test the sensor, I placed it inside the fridge to test if it works well under cold conditions (it's datasheet states that it should work from -40C to 125C). The readings do reflect the expected output (lower temperature and humidity)

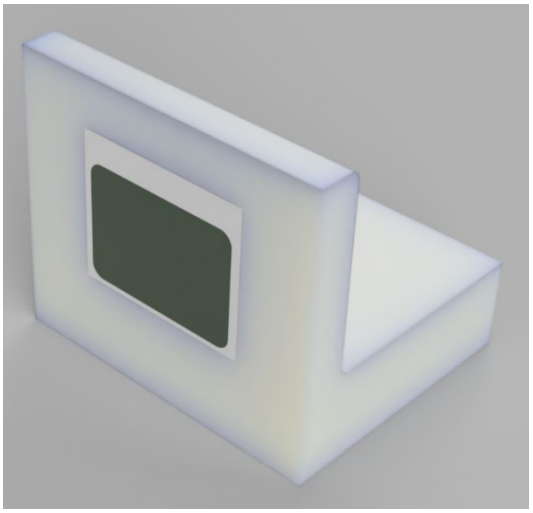
STEP 6: REAL TIME CLOCK

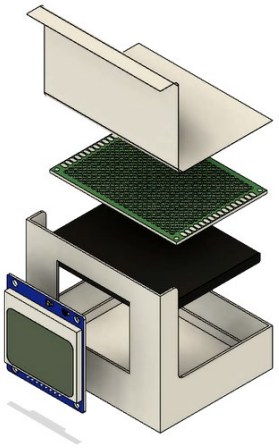
The real time clock I purchased does not come with a battery and must be constantly powered by the Arduino. This means that any loss of power will cause the time to reset to its original time. Furthermore, it is power-intensive as well, 10 times more than what I expected. After doing some research online, I found out that the model I purchased comes with a battery charging IC, which draws power, even on standby mode. This will cause a 50% increase in power draw. Hence, I will not be including the Real Time Clock into the final product. I proceeded to reorganise the screen layout to make better user of the avaliable space.



CONSTRUCTION OF ENCLOSURE

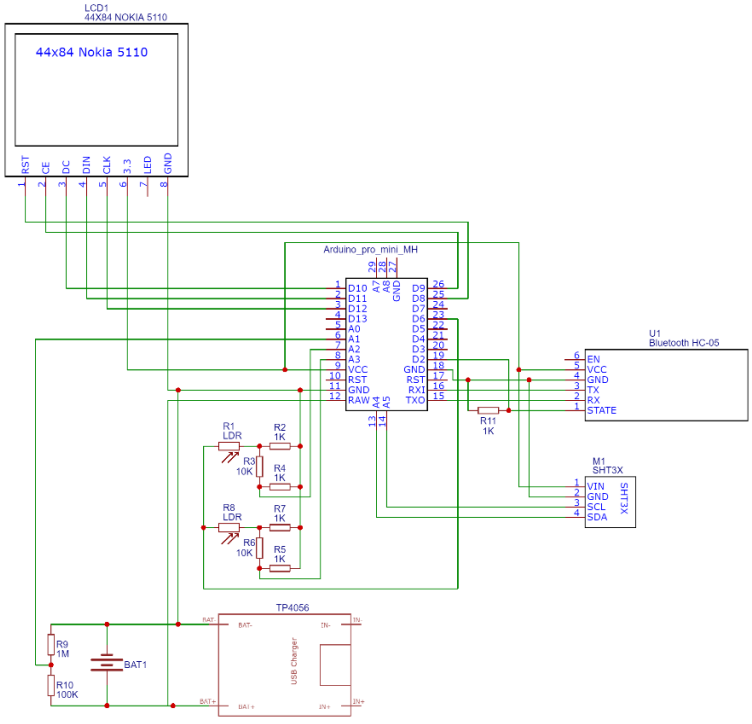
I started by measuring the size of the items, mainly: the battery and the PCB board. With the dimensions, I created an enclosure which can fit the battery, followed by the PCB Board. After which, I extended the front of the enclosure to make space for the display. As this design is meant to be 3D printed, I could not add a roof. Hence, I will be using chiyogami paper to cover the roof. This will improve the aesthetics and keep the electronics hidden at the same time.

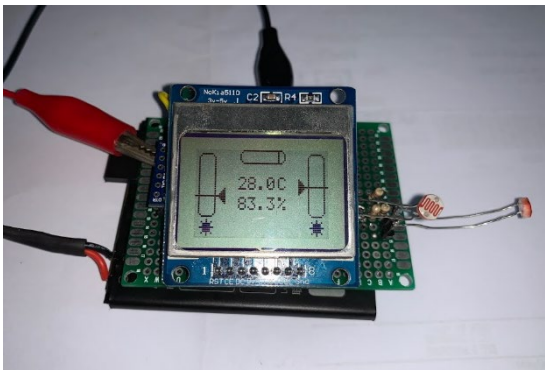


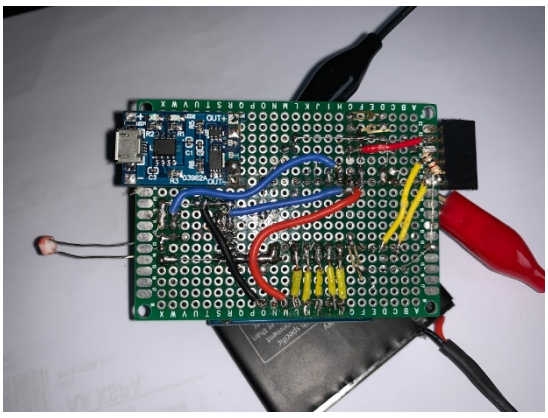


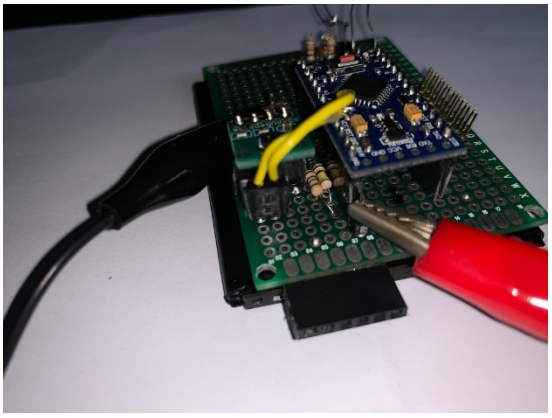
Exploded View Video: https://youtu.be/O5aU29klYh0

After designing the case, I proceeded to build the circuit, which reduces the amount of space needed. Firstly, using the breadboard circuit as a reference, I created a schematic diagram, so that I know how to wire the components together. Then, I proceeded to transfer the components from the breadboard to the Perf Board and soldering the connections together.



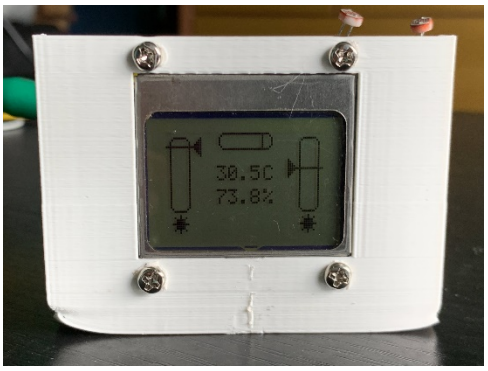






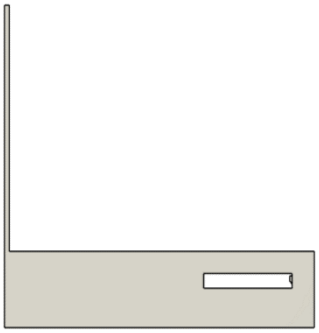
FIRST 3D PRINT

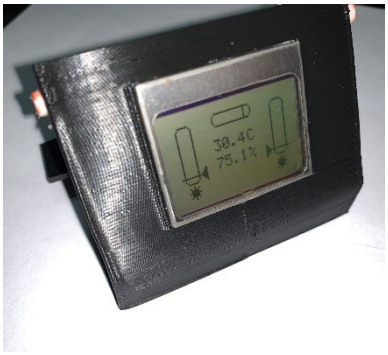
The first 3D print has some significant errors. Firstly, the corners of the base warped in on itself. The compartment for the battery is also too small, making it impossible for me to fit the battery inside. Finally, I realised that it is difficult to see the display when it is perpendicular to the table. A display that is tilted at an angle, like a laptop screen to improve the visibility. I have explored various options and decided that the most efficient way was to bend the display portion of the 3D print to the desired angle.



SECOND 3D PRINT

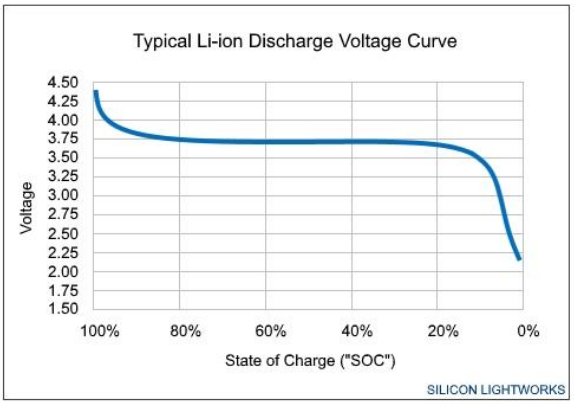
I reduced the thickness of the part which holds the screen, which will allow it to be bent to a suitable angle. This is done by using a heat gun to heat up the print until it is bendable and tilting it to the desired position. After which, I followed by soldering the connectors from the LCD to the breadboard. This is followed by the assembly of battery and LDR. I tested the Bluetooth and Charging feature, and both worked.



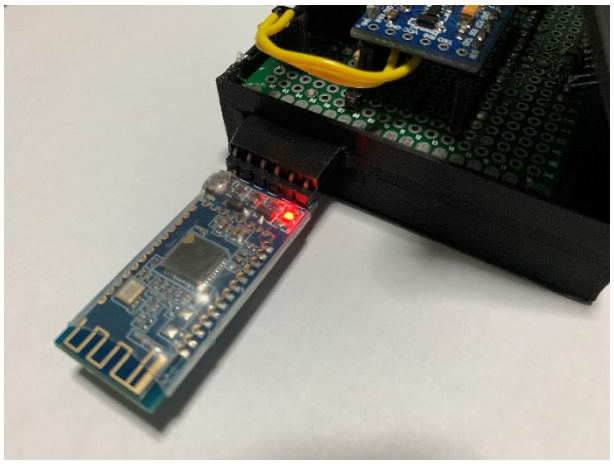


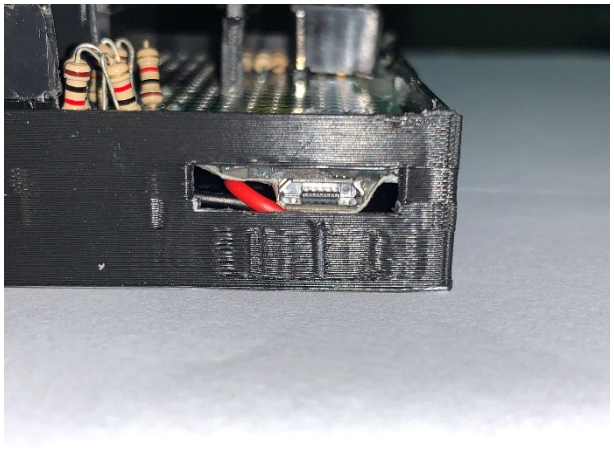
OBSERVATIONS

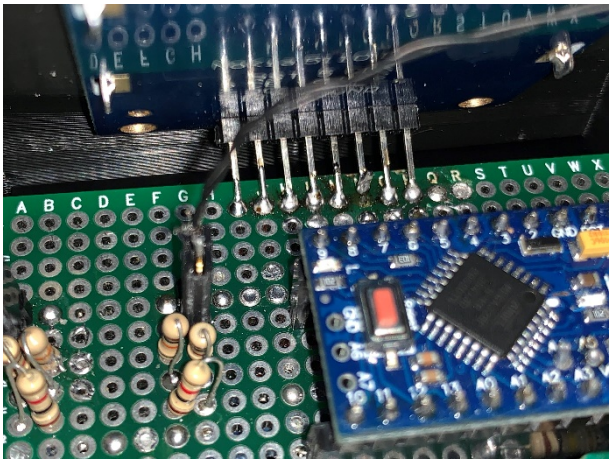
I have run the weather system for 3 days and the battery voltage dropped from 4V to 3.94V. Using a battery percentage graph, I can estimate about 2% of battery drain. This means, it can be powered for more than 2 months, which is consistent with my calculations. I have also made some interesting observations in the temperature and humidity readings. Firstly, the relative humidity will be higher (>80%) during the night and lower (60%) during the day, unless it is raining. When it is raining, the relative humidity of the room can increase to 90%. Furthermore, the temperature will increase from 27C during the night to 32C during the day.

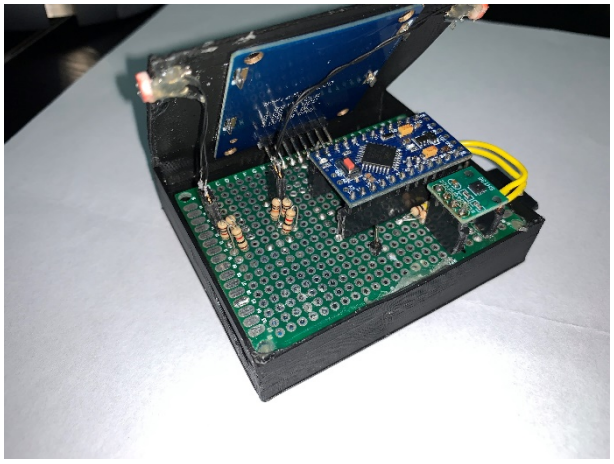


COMPLETED









LEARNING POINTS

1. Different types of batteries and their advantages/disadvantages
    a. Shape
    b. Energy Density
    c. Internal Resistance
    d. Lifespan
    e. Recharge speed
2. Arduino C coding
    a. Using of functions
    b. Using of library
    c. Lowering clock speed to improve battery life
    d. Using power efficient library
3. 3D design and printing
    a. Building a 3D model which can print easily, with minimum supports
4. PCB design and perf board building
    a. Planning of design
    b. Soldering
    c. Building of connectors
    d. Diagnosing PCB that is not working=

CODES

```
#include <LCD5110_Graph.h>
#include "Adafruit_SHT31.h"
#include <SPI.h>
#include <LowPower.h>

#define battery A1
#define ldrL A3
#define ldrR A2
#define SDAPIN  A4
#define SCLPIN  A5
#define btSTATE 2
#define lightEnable 9
#define iREF 1.09 //internal reference cal factor
#define multiplier 11.83
#define battMin 3.3

LCD5110 lcd(7, 6, 5, 3, 4);

extern unsigned char SmallFont[];
char temperature[6];
char humidity[6];

const unsigned char brightness [] PROGMEM = {
  0x10, 0x92, 0x7c, 0x7c, 0xff, 0x7c, 0x7c, 0x92, 0x10, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00,
  0x00, 0x00
};

const unsigned char arrowL [] PROGMEM = {
  0x10, 0x38, 0x7c, 0xfe, 0xff, 0x00, 0x00, 0x00, 0x00, 0x01
};

const unsigned char arrowR [] PROGMEM = {
  0xff, 0xfe, 0x7c, 0x38, 0x10, 0x01, 0x00, 0x00, 0x00, 0x00
};

const unsigned char bt [] PROGMEM = {
  0x04, 0x88, 0x50, 0xff, 0x22, 0x54, 0x88, 0x01, 0x00, 0x00, 0x07, 0x02, 0x01, 0x00
};


float maxLight = 0;
```

```
bool state = 0;

Adafruit_SHT31 sht35 = Adafruit_SHT31();

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

 lcd.InitLCD();
 lcd.setFont(SmallFont);

 pinMode(lightEnable, OUTPUT);
 pinMode(ldrL, INPUT);
 pinMode(ldrR, INPUT);
 pinMode(battery, INPUT);
 pinMode(btSTATE, INPUT);
 btChange();

 attachInterrupt(digitalPinToInterrupt(btSTATE), btChange, CHANGE);

 Wire.begin();
 sht35.begin(0x44);
}

void loop() {
 screen0();
 updateTempHumi();
 updateBattery();
 updateLight();
 if (state) {
  lcd.drawBitmap(39, 37,  bt, 7, 11);
  lcd.update();
  delay(50);
  LowPower.idle(SLEEP_1S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF, SPI_OFF, USART0_OFF, TWI_OFF);
  Serial.println();
 }
 else {
  lcd.update();
  delay(50);
  LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF, SPI_OFF, USART0_OFF, TWI_OFF);
 }
}

void btChange() {
 state = digitalRead(btSTATE);
}


void updateBattery() {
 float battVolt = analogRead(battery) * multiplier / 1000;
 if (battVolt >= battMin) {
  int battPercent = (battVolt - battMin) / (4 - battMin) * 24;
  if (battPercent > 24) {
   battPercent = 24;
  }

  if (state) {
   Serial.print("Battery = ");
   Serial.print(battVolt);
   Serial.print("V | ");
   Serial.print((battVolt - battMin) / (4 - battMin) * 100);
   Serial.println("%");
  }
  lcd.drawLine(30 + battPercent, 0, 30 + battPercent , 7);
 }
 else {
  lcd.print("NO BATT", 20, 0);
 }
```

```
}

void updateLight() {
  digitalWrite(lightEnable, HIGH);
  delay(10);
  float sensorValueL = 0;
  float sensorValueR = 0;
  for (int x = 0; x < 10; x++) {
    sensorValueL += analogRead(ldrL);
    sensorValueR += analogRead(ldrR);
  }
  sensorValueL = sensorValueL / 10;
  sensorValueR = sensorValueR / 10;
  digitalWrite(lightEnable, LOW);

  if (sensorValueL > maxLight) {
    maxLight = sensorValueL;
  }
  if (sensorValueR > maxLight) {
    maxLight = sensorValueR;
  }
  int lightValueL = (sensorValueL / maxLight * 30);
  int lightValueR = (sensorValueR / maxLight * 30);
  if (state) {
    Serial.print("Light (L) = ");
    Serial.print(sensorValueL / maxLight * 100);
    Serial.println("%");

    Serial.print("Light (R) = ");
    Serial.print(sensorValueR / maxLight * 100);
    Serial.println("%");
  }


  lcd.drawLine(0, 34 - lightValueL, 15, 34 - lightValueL);
  lcd.drawLine(69, 34 - lightValueR, 84, 34 - lightValueR);

  lcd.drawBitmap(14, 30 - lightValueL,  arrowL, 5, 9);
  lcd.drawBitmap(65, 30 - lightValueR,  arrowR, 5, 9);
}

void updateTempHumi() {
  float t = sht35.readTemperature();
  float h = sht35.readHumidity();

  if (! isnan(t)) {  // check if 'is not a number'
    if (state) {
      Serial.print("Temperature = "); Serial.print(t); Serial.println("C");
    }

    dtostrf(t, 3, 1, temperature);
    lcd.print(temperature, 26, 15);
    lcd.print("C", 51, 15);
  } else {
    lcd.print("NC", 36, 15);
  }

  if (! isnan(h)) {  // check if 'is not a number'
    if (state) {
      Serial.print("Humidity = "); Serial.print(h); Serial.println("% ");
    }

    dtostrf(h, 3, 1, humidity);
    lcd.print(humidity, 26, 26);
    lcd.print("%", 51, 26);
  } else {
    lcd.print("NC", 36, 26);
  }
```

```
}


void screen0() {
  lcd.clrScr();
//  lcd.drawLine(42, 0, 42, 47);
  lcd.drawRoundRect(2, 2, 12, 36);
  lcd.drawRoundRect(71, 2, 81, 36);
  lcd.drawRoundRect(28, 0, 56, 7);
  lcd.drawBitmap(3, 38,  brightness, 9, 9);
  lcd.drawBitmap(72, 38,  brightness, 9, 9);
}
```