**PUBLIC TRANSPORTATION OPTIMIZATION**

Creating a platform for battery percentage checking using MIT App Inventor involves several steps. MIT App Inventor is a user-friendly platform for creating Android apps, and it can be used to build a mobile app that monitors and controls water levels in a tank. Here's a basic outline of the steps involved.

**Block-Based Coding:**

• Use MIT App Inventor's block-based coding to program the app:

• When Screen1.Initialize:

• Initialize any necessary variables .

• When Button.Click:

• Update the status label accordingly .

• Use a Timer component to periodically check the battery percentage (simulated for this example).

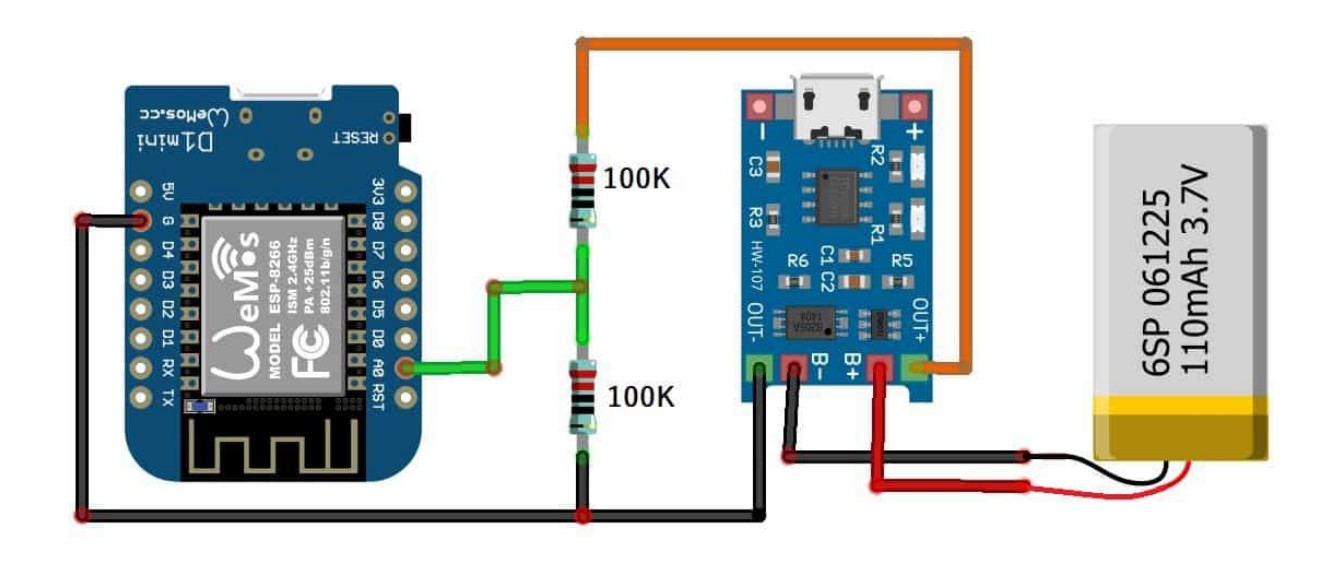
• Display the battery percentage on the appropriate label.

• Add logic to check if the battery percentage exceeds a predefined threshold.

• If the battery percentage is above the threshold, change the status label to "Charging Alert!" and send a notification to the user.

• To simulate the notification, you can use the Notifier component to show an alert message when the threshold is crossed.

**Circuit design & program:**

****

**Input Program:**

#include <ESP8266WiFi.h>

#designwrite FIRE\_BASE URL "https://pto-17f09-default-rtdb.firhttps://pto-17f09-default-rtdb.firebas/ebaseio.com"

#designwrite FIRE\_BASE KEYWORD "https://pto-17f09-default-rtdb.firebaseio.com"

String apiKey = "\*\*\*\*\*\*";

const char\* ssid = "\*\*\*\*\*\*"; // Enter your WiFi Network's SSID

const char\* pass = "\*\*\*\*\*\*"; // Enter your WiFi Network's Password

const char\* server = "api.thingspeak.com";

int analogInPin = A0; // Analog input pin

int sensorValue; // Analog Output of Sensor

float calibration = 0.36; // Check Battery voltage using multimeter & add/subtract the value

int bat\_percentage;

WiFiClient client;

void setup()

{

Serial.begin(115200);

Serial.println("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, pass);

while (WiFi.status() != WL\_CONNECTED)

{

delay(100);

Serial.print("\*");

}

Serial.println("");

Serial.println("WiFi connected");

}

void loop()

{

sensorValue = analogRead(analogInPin);

float voltage = (((sensorValue \* 3.3) / 1024) \* 2 + calibration); //multiply by two as voltage divider network is 100K & 100K Resistor

bat\_percentage = mapfloat(voltage, 2.8, 4.2, 0, 100); //2.8V as Battery Cut off Voltage & 4.2V as Maximum Voltage

if (bat\_percentage >= 100)

{

bat\_percentage = 100;

}

if (bat\_percentage <= 0)

{

bat\_percentage = 1;

}

Serial.print("Analog Value = ");

Serial.print(sensorValue);

Serial.print("\t Output Voltage = ");

Serial.print(voltage);

Serial.print("\t Battery Percentage = ");

Serial.println(bat\_percentage);

delay(1000);

if (client.connect(server, 80))

{

String postStr = apiKey;

postStr += "&field1=";

postStr += String(voltage);

postStr += "&field2=";

postStr += String(bat\_percentage);

postStr += "\r\n\r\n";

client.print("POST /update HTTP/1.1\n");

delay(100);

client.print("Host: api.thingspeak.com\n");

delay(100);

client.print("Connection: close\n");

delay(100);

client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");

delay(100);

client.print("Content-Type: application/x-www-form-urlencoded\n");

delay(100);

client.print("Content-Length: ");

delay(100);

client.print(postStr.length());

delay(100);

client.print("\n\n");

delay(100);

client.print(postStr);

delay(100);

}

client.stop();

Serial.println("Sending....");

delay(15000);

}

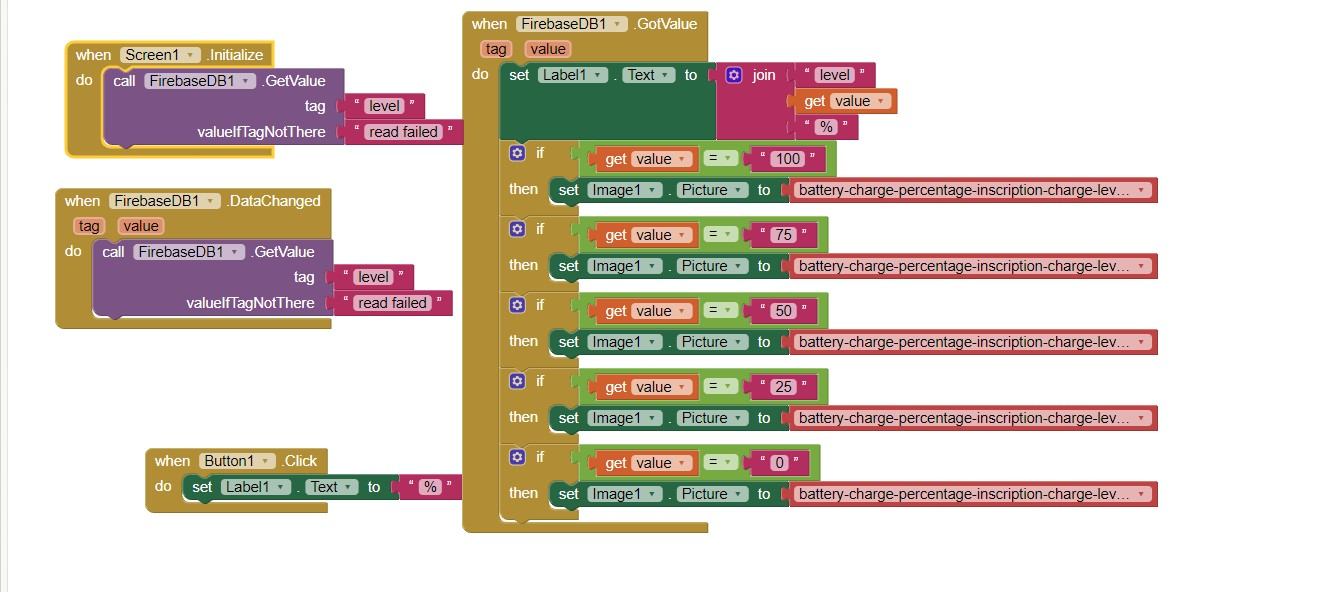
float mapfloat(float x, float in\_min, float in\_max, float out\_min, float out\_max)

{

return (x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + out\_min;

}

**Block design for app:**

****

**Lithium-Ion Batteries:**

A **lithium-ion** battery or **Li-ion** battery is a type of **rechargeable battery**. Lithium-ion batteries are commonly used for portable electronics and electric vehicles.

In this battery, lithium ions move from the **negative electrode** through an electrolyte to the **positive electrode** during discharge, and back when charging. Li-ion batteries use an intercalated lithium compound as the material at the positive electrode and typically **graphite** at the negative electrode. The batteries have a high energy density, no memory effect and low self-discharge.

**Nominal, Maximum & Cut-off Voltage:**

I’ve been using several Lithium-Ion batteries for quite some time across multiple projects. Some of these batteries come with an attached Battery Management System Circuit, providing over-voltage protection, balanced charging, and short-circuit protection.

Typically, Lithium-Ion batteries have a nominal voltage of 3.7V. When fully charged, their maximum voltage can reach up to 4.2±0.5V. Manufacturer datasheets usually state the cut-off voltage to be around 3V, though this can vary based on the battery type and its specific applications. The battery I frequently use has a discharge cut-off voltage of 2.8V. However, there are also batteries available with a cut-off voltage as low as 2.5V.

**Circuit & Schematic Designing:**

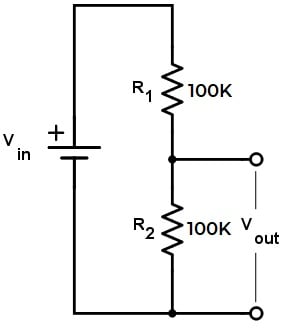
We will design a system to monitor this battery voltage along with charging and discharging status. For the microcontroller, we use **Wemos D1 Mini** which has an **ESP8266** wifi-enabled chip. You can also use the **NodeMCU ESP8266 Board**. This WiFi chip can connect to the WiFi network and uploads the data regularly to the server.

You can use the TP4056  module to charge the battery as its best suited for Battery Management Applications. The MCP73831 IC can also be used instead of TP4056.

The ESP8266 Chip can only support the input analog voltage of **3.3V**. But Battery voltage goes up to **4.2V**. Hence we have to form a voltage divider network to lower down the input voltage.

**Voltage Divider Network Calculations:**

The Battery Maximum voltage is 4.2V and the cut of voltage is 2.8V. Anything lesser than 3.3V will be easily supported by ESP8266 Analog Pin.



We have to first step down the upper voltage level. The **source voltage** is **4.2V** and there is a pair of **100K resistors**. This will give an output of **2.1V**. Similarly, the **minimum voltage** is **2.8V** as a cutoff voltage which steps down to 1.4V using the same voltage divider network. Hence, both the upper and lower voltage is supported by **ESP8266 Analog Pin**.