Naan Mudhalvan Phase 3 Assessment

Course Name : Internet Of Things

Project Title : Public Transportation Optimization

Team Name : TechSpark

Team Members :

|  |  |  |
| --- | --- | --- |
| Name | Register number | Nan Mudhalvan ID |
| Praveen U.S | 721221106069 | au721221106069 |
| Sanjay S | 721221106088 | au721221106088 |
| Sankar Maharajan G | 721221106090 | au721221106090 |
| Murugesan | 72I221106303 | au2272120025 |

Simulator used :  **Tinkercad Simulator .**

Components used :

|  |  |  |
| --- | --- | --- |
| Component Name | Specifications | Quantity |
| Arduino Board | Uno R3 | 2 |
| GPS Module | Tiny GPS | 1 |
| DHT Sensor | - | 1 |
| LCD Display | 16 x 2 | 2 |
| Resistor | 220 Ω , 330 Ω | Each 1 |
| Power Source Cable | - | 2 |

Libraries Used :

1 . RPi. GPIO Library ( Rasperry pi )

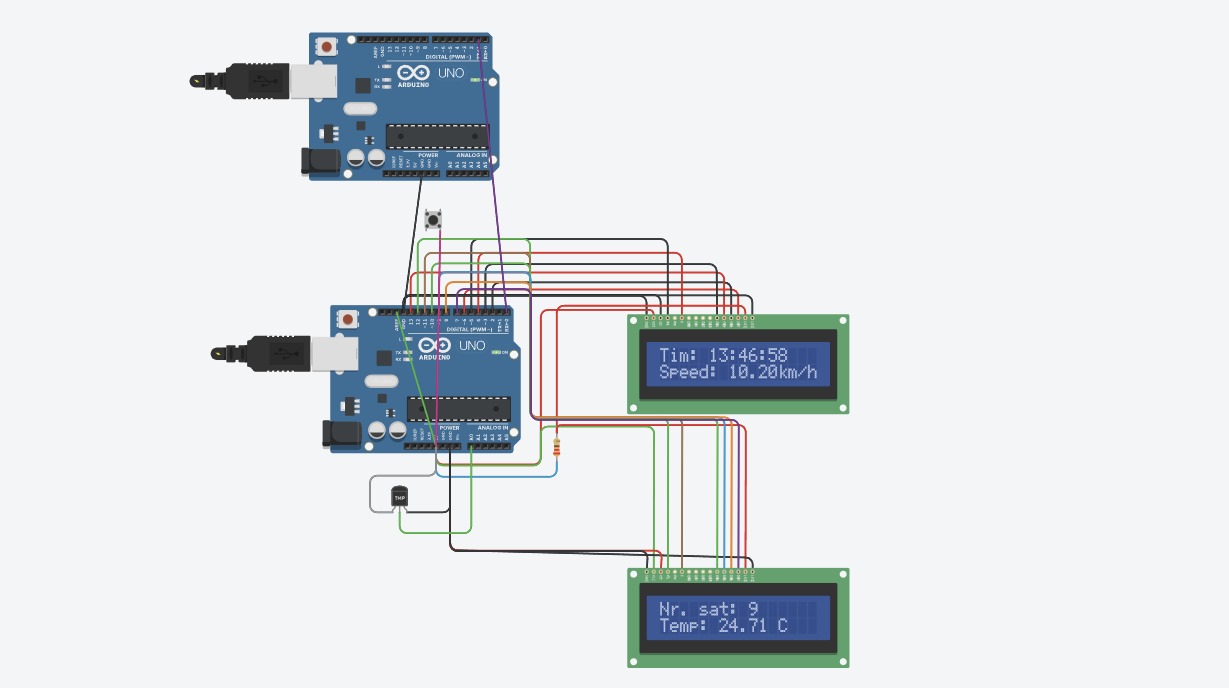
2. Tiny Gps ++ Library

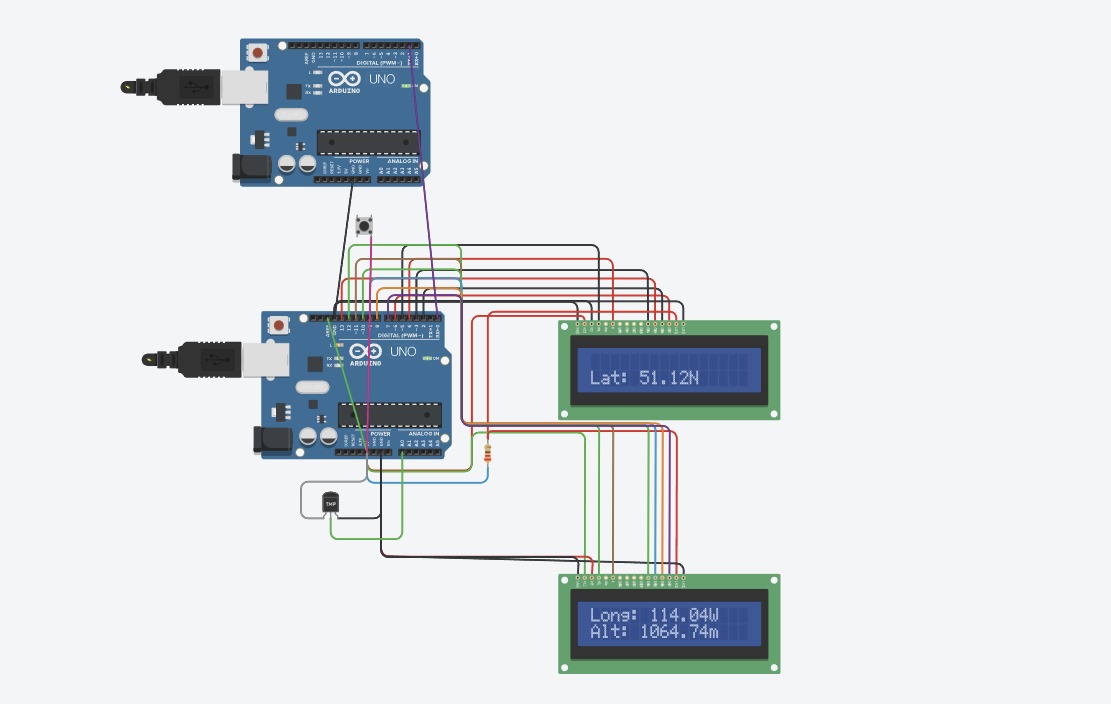
3. Adafruit \_SSD1306

4. DHT Library

5. Wire Library

**Circuit Diagram :**





**Source Code :**

import time

import serial

import RPi.GPIO as GPIO

# Define the GPIO pins for the LCD

lcd\_rs = 12

lcd\_en = 11

lcd\_d4 = 5

lcd\_d5 = 4

lcd\_d6 = 3

lcd\_d7 = 2

# Set up the GPIO mode and pin setup

GPIO.setmode(GPIO.BCM)

GPIO.setup(lcd\_rs, GPIO.OUT)

GPIO.setup(lcd\_en, GPIO.OUT)

GPIO.setup(lcd\_d4, GPIO.OUT)

GPIO.setup(lcd\_d5, GPIO.OUT)

GPIO.setup(lcd\_d6, GPIO.OUT)

GPIO.setup(lcd\_d7, GPIO.OUT)

# Define the LCD column and row size

lcd\_columns = 16

lcd\_rows = 2

# Create functions for sending data to the LCD

def lcd\_byte(bits, mode):

GPIO.output(lcd\_rs, mode)

GPIO.output(lcd\_d4, False)

GPIO.output(lcd\_d5, False)

GPIO.output(lcd\_d6, False)

GPIO.output(lcd\_d7, False)

if bits & 0x10 == 0x10:

GPIO.output(lcd\_d4, True)

if bits & 0x20 == 0x20:

GPIO.output(lcd\_d5, True)

if bits & 0x40 == 0x40:

GPIO.output(lcd\_d6, True)

if bits & 0x80 == 0x80:

GPIO.output(lcd\_d7, True)

lcd\_toggle\_enable()

GPIO.output(lcd\_d4, False)

GPIO.output(lcd\_d5, False)

GPIO.output(lcd\_d6, False)

GPIO.output(lcd\_d7, False)

if bits & 0x01 == 0x01:

GPIO.output(lcd\_d4, True)

if bits & 0x02 == 0x02:

GPIO.output(lcd\_d5, True)

if bits & 0x04 == 0x04:

GPIO.output(lcd\_d6, True)

if bits & 0x08 == 0x08:

GPIO.output(lcd\_d7, True)

lcd\_toggle\_enable()

def lcd\_toggle\_enable():

time.sleep(0.0005)

GPIO.output(lcd\_en, True)

time.sleep(0.0005)

GPIO.output(lcd\_en, False)

time.sleep(0.0005)

def lcd\_string(message):

message = message.ljust(lcd\_columns, " ")

for i in range(lcd\_columns):

lcd\_byte(ord(message[i]), GPIO.HIGH)

# Open a serial port for communication with the GPS

ser = serial.Serial('/dev/ttyS0', 9600, timeout=1)

# Set up the LCD display

GPIO.output(lcd\_rs, GPIO.LOW)

lcd\_byte(0x33, GPIO.LOW)

lcd\_byte(0x32, GPIO.LOW)

lcd\_byte(0x28, GPIO.LOW)

lcd\_byte(0x0C, GPIO.LOW)

lcd\_byte(0x06, GPIO.LOW)

lcd\_byte(0x01, GPIO.LOW)

try:

while True:

gps\_data = ser.readline().decode('utf-8')

print("Data from GPS:")

print(gps\_data)

# Extract LAT, LOG, VEL from GPS data as needed

LAT = 0x10

LOG = 0x20

VEL = 0x30

lcd\_string("Lat:{} Long:{} Vel:{}".format(LAT, LOG, VEL))

time.sleep(1)

except KeyboardInterrupt:

pass

finally:

ser.close()

GPIO.cleanup()

**JSON code :**

import time

import board

import digitalio

import busio

# Define the LED pin

led = digitalio.DigitalInOut(board.D13)

led.direction = digitalio.Direction.OUTPUT

# Create a software serial port

uart = busio.UART(board.TX, board.RX, baudrate=9600)

values = [

0x20, 0x40, 0x42, 0x41, 0x13, 0x10, 0x60, 0x42,

0x60, 0x00, 0x00, 0x41, 0x15, 0x10, 0x55, 0x42,

0x00, 0x00, 0x60, 0x42, 0x41, 0x15, 0x20, 0x45,

0xB0, 0x00, 0x00, 0x41

]

while True:

for value in values:

uart.write(bytearray([value]))

led.value = True # Turn on the LED

time.sleep(0.5)

led.value = False # Turn off the LED

time.sleep(0.5)

**Working :**

In this smart transportation optimization project, we've integrated a range of components to create an efficient and data-driven system. At the core of our setup, an Arduino board serves as the central processing unit, orchestrating the flow of data and control throughout the system. The Arduino communicates with a GPS module, allowing us to track the real-time location of vehicles and passengers, ensuring accurate routing and timely arrivals. To monitor environmental conditions, a DHT sensor is employed, collecting data on temperature and humidity for enhanced route optimization. The LCD display provides drivers and passengers with instant access to route information and important notifications. A resistor ensures stable voltage and signal levels, while a reliable power source cable keeps the system operational. Together, these components form the backbone of our smart transportation solution, delivering improved efficiency, safety, and user experience.