SCHOOL OF COMPUTING (SOC)

IOT CA2
Step-by-step Tutorial

DIPLOMA IN BUSINESS INFORMATION TECHNOLOGY
DIPLOMA IN INFORMATION TECHNOLOGY
DIPLOMA IN INFOCOMM SECURITY MANAGEMENT

ST0324 Internet of Things (IOT)

Date of Submission: 23 Feb 2020

Prepared for: Ms Dora

Class: DIT/FT/2B/34

Submitted by:

Student ID	Name
1846742	Ang Wei Bo Gary
1828881	Muhammad Darius Dani Bin Adil Choo
1828513	Edward Lin

Table of Contents

Section 1
Overview of project

A. Where we have uploaded our tutorial

Fill up the Google form here to submit your links and then paste the links here of your Youtube and tutorial document here as well.

http://bit.ly/1910s2iotca2

Youtube	https://www.youtube.com/watch?v=QcsE--hGc_s
Public tutorial link	https://github.com/EdwardBrick/IOTCA2byIDK.git

B. What is the application about?

Login of ATS using RFID card. Student can log their attendance through tapping of card on the device inside the classroom. The device aims to serve both staff and students. As for the lecturers, they can use it to track their students' attendance.
The device cuts down on the number of ATS failure in school when the wifi is down or weak.

C. How does the final RPI set-up looks like?

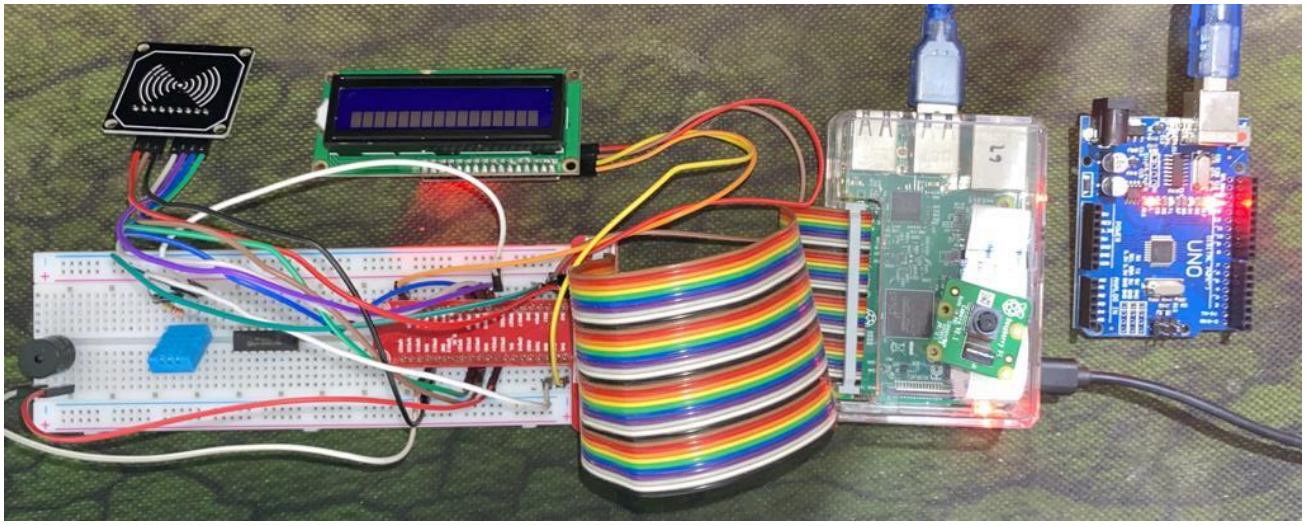


Figure 1 Overall View

This will be in the every classroom.

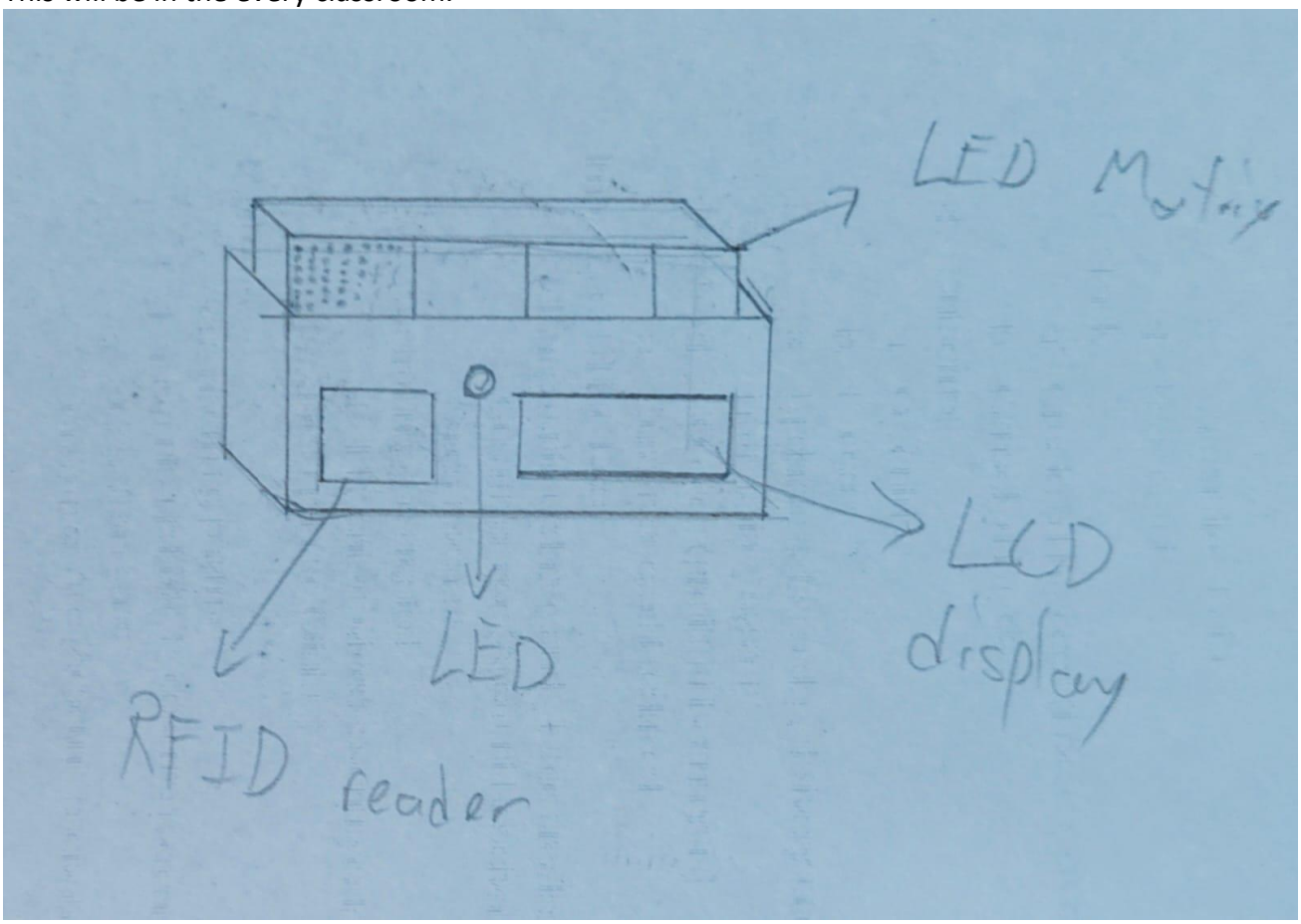
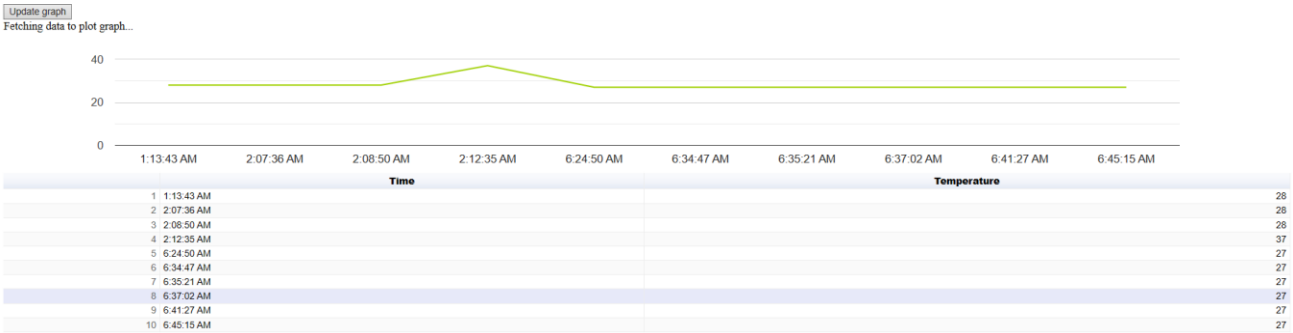


Figure 2 Covers to protect the device

D. How does the web or mobile application look like?



	Time	Temperature
1	1:13:43 AM	28
2	2:07:36 AM	28
3	2:08:50 AM	28
4	2:12:35 AM	37
5	6:24:50 AM	27
6	6:34:47 AM	27
7	6:35:21 AM	27
8	6:37:02 AM	27
9	6:41:27 AM	27
10	6:45:15 AM	27

Figure 3 Webpage

Provide at least one screenshot of your web app, and more if your web app consists of more than 1 page. Otherwise, I will assume your webapp only can show 1 page. Label your screenshots so that they may be referenced in Section F.

E. System architecture of our system

Provide a hand-drawn or computer-drawn system architecture diagram please. Example given below.

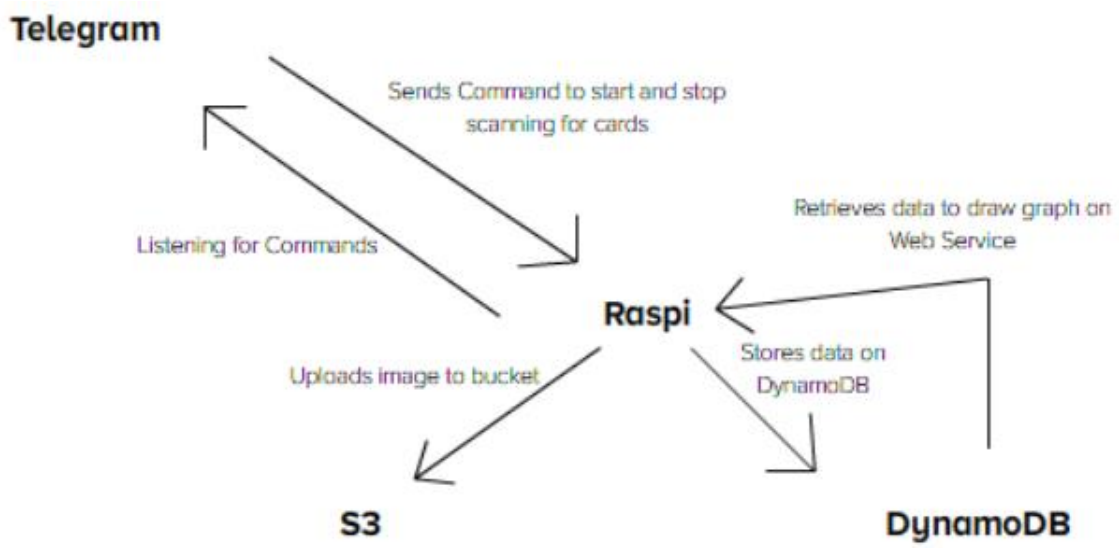


Figure 4 Our System

F. Evidence that we have met basic requirements

Provide bullet list to describe how your group has met basic requirements

Requirement	Evidence
Used three sensors	Used Camera and RFID sensor. Also used DHT11
Used MQTT	Our MQTT endpoint --> amitv187sde3m-ats.iot.us-east-1.amazonaws.com Example of data sent through MQTT : {"datetimeid": "2020-02-22T01:13:43.534875", "deviceid": "deviceid_dariuschoo", "value": "New card detected! UID of card is [136, 4, 206, 68, 6]", "temperature": "28"}
Stored data in cloud	Stored RFID data in Cloudant database in dynamoDB cloud
Used cloud service	Use S3, Store Images
Provide real-time sensor value / status	Show the real-time of taking attendance, displaying time on the LED Matrix
Provide historical sensor value/ status	Show historical vlaue of DHT11 sensor
Control actuator	Used telepot bot to turn off and on the RFID scanner

G. Bonus features on top of basic requirements

Provide bullet list of the bonus features you have added on top of basic requirements

a) Used Clock Module paired with LED Matrix and Arduino to print out current time

A. Quick-start guide (Readme first)

1) First connect hardware as in Section 2
2) Install Library needed given in Section 3
3) Code RFIDWorks.py and Server.py
4) Code Arduino codes using Arduino IDE
5) Run RFIDWorks.py first
6) Then run the Server.py file for web server

Section 2
Hardware requirements

Hardware checklist

NFC / RFID Card Reader
DHT11
Buzzer
I2C LCD Screen
4in1 max7219 LED Matrix
Arduino Uno
PiCamera
10k Ω Resistor
DS1307 Clock Module
Raspberry Pi 3 Model B

Hardware setup instructions

Setting up one by one as shown in the fritzing diagram

Fritzing Diagram

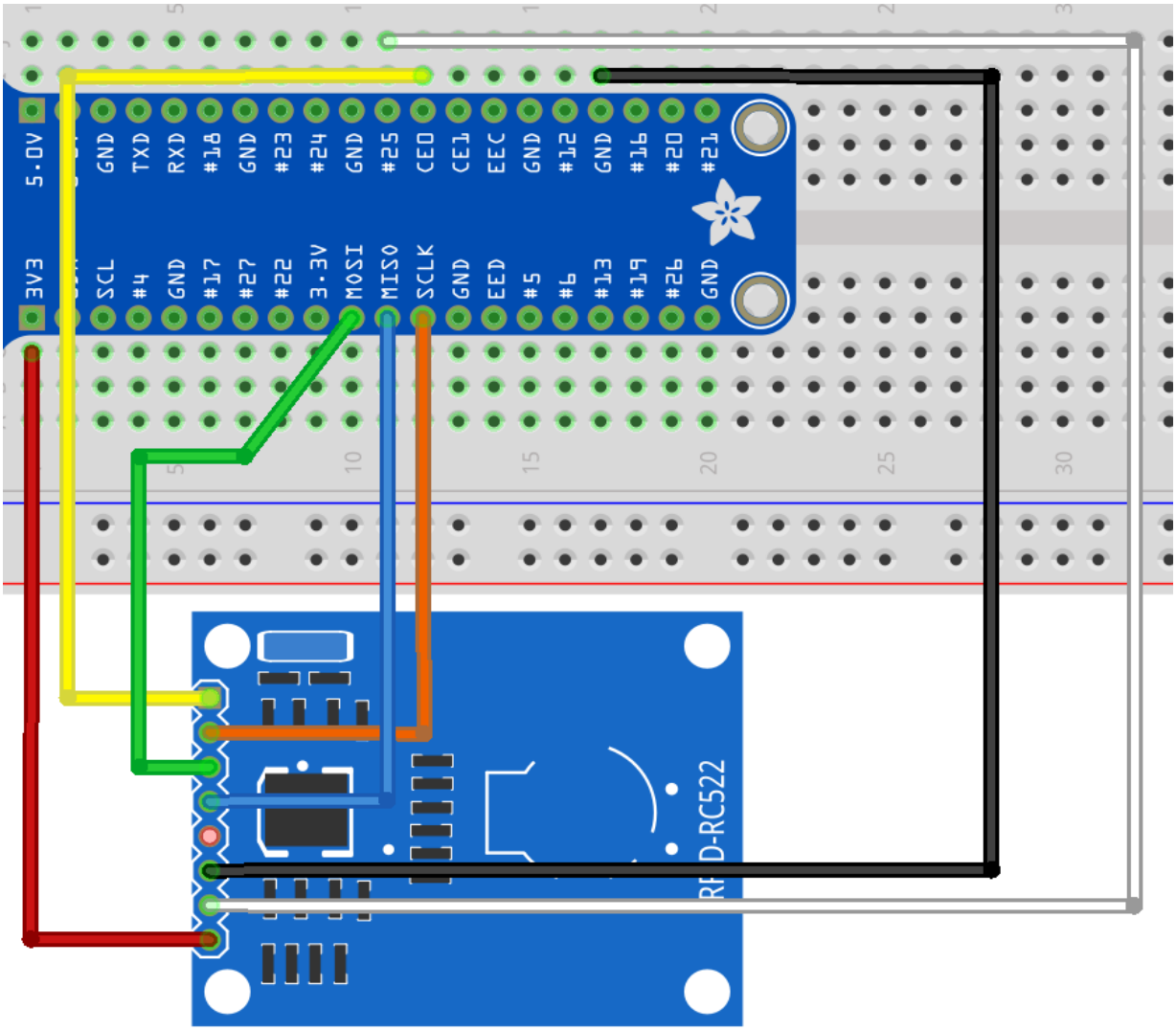


Figure 5 RFID

Setting up RDIF First

RFID

Jumper color	MFRCF522pin	RPi pin
Yellow	SDA	CE0
Orange	SCK	SCLK
Green	MOSI	MOSI
Blue	MISO	MISO
	IDR	
Black	GND	GND
White	RST	GPIO25
Red	3.3V	3.3V
	5V	

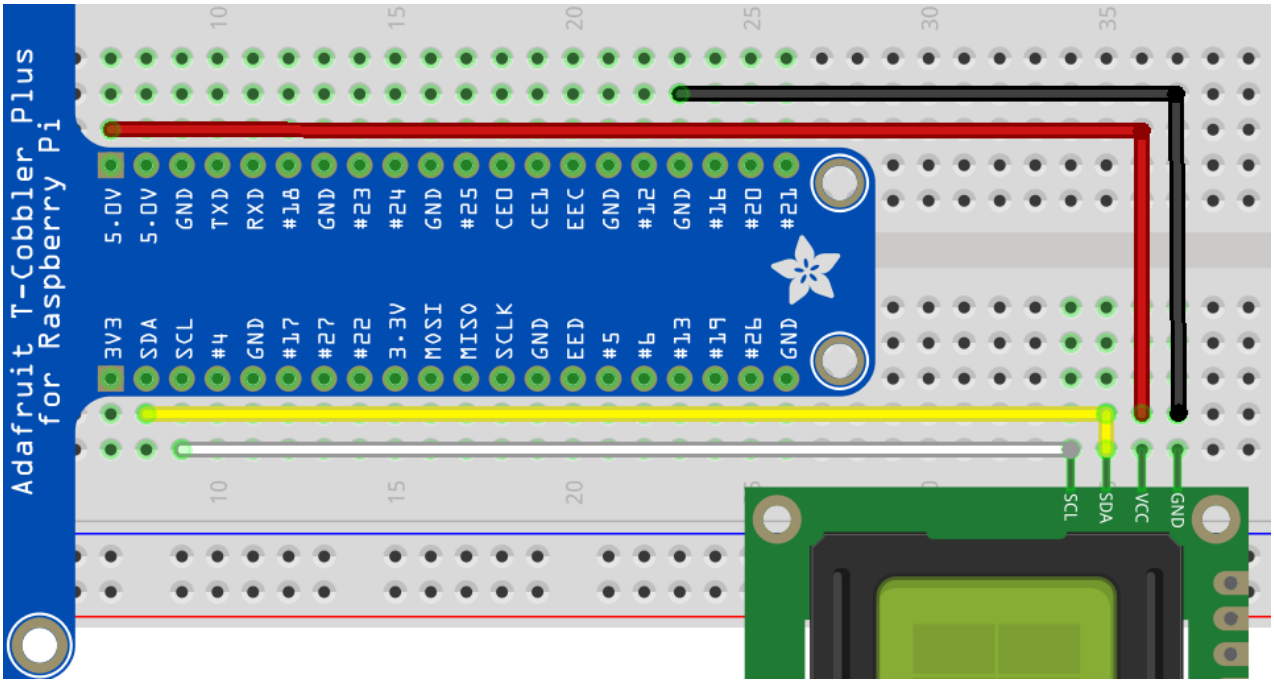


Figure 6 LCD

Setting up the LCD as Followed

LCD

Jumper color	LCD pin	RPi pin
White	SCL	SCL
Yellow	SDA	SDA
Black	GND	GND
Red	Vcc	5V

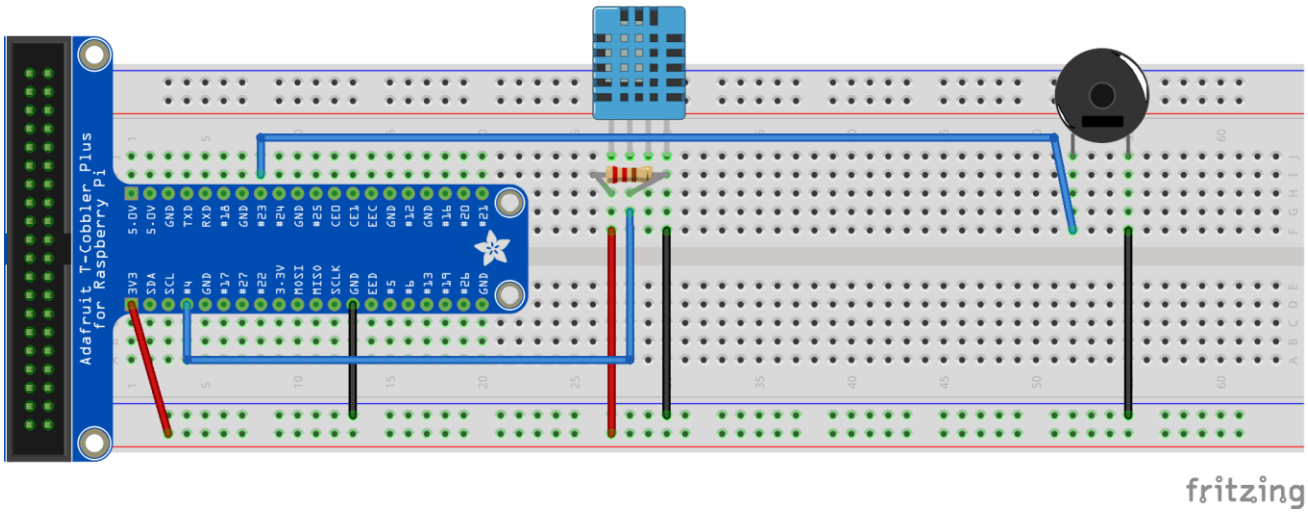


Figure 7 DHT and Buzzer

Setting DH11 and Buzzer

DHT11

Jumper color	DHT11 pin	RPi pin
Red	3.3V	3.3V
Blue	DATA	GPIO4 / BCM4
Black	GND	GND

Buzzer

Jumper color	Buzzer pin	RPi pin
Blue	DATA	GPIO4 / BCM4
Black	GND	GND

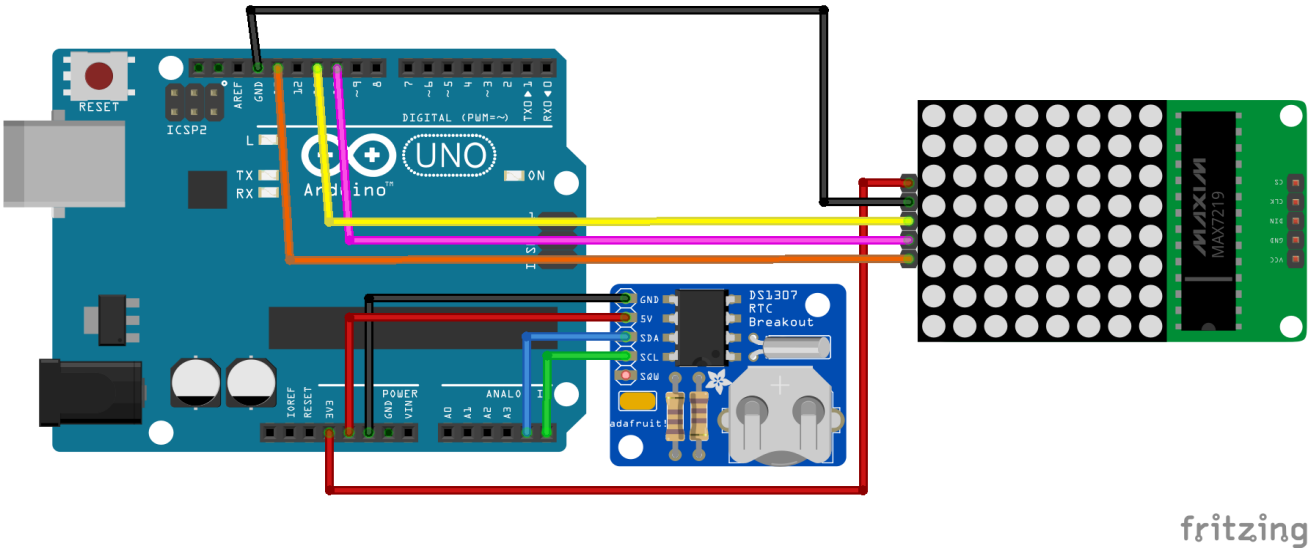


Figure 8 Clock Module and LED

Lastly Clock Module and LED

4in1 max7219 LED

Jumper color	LED pin	RPi pin
Red	3.3V	3.3V
Black	GND	GND
Yellow	DIN	11
Pink	CS	10
Orange	CLK	13

Clock Module DS1307

Jumper color	DS1307 pin	RPi pin
Black	GND	GND
Red	5V	5V
Blue	SDA	A4
Green	SCL	A5

Fritzing Diagram Final

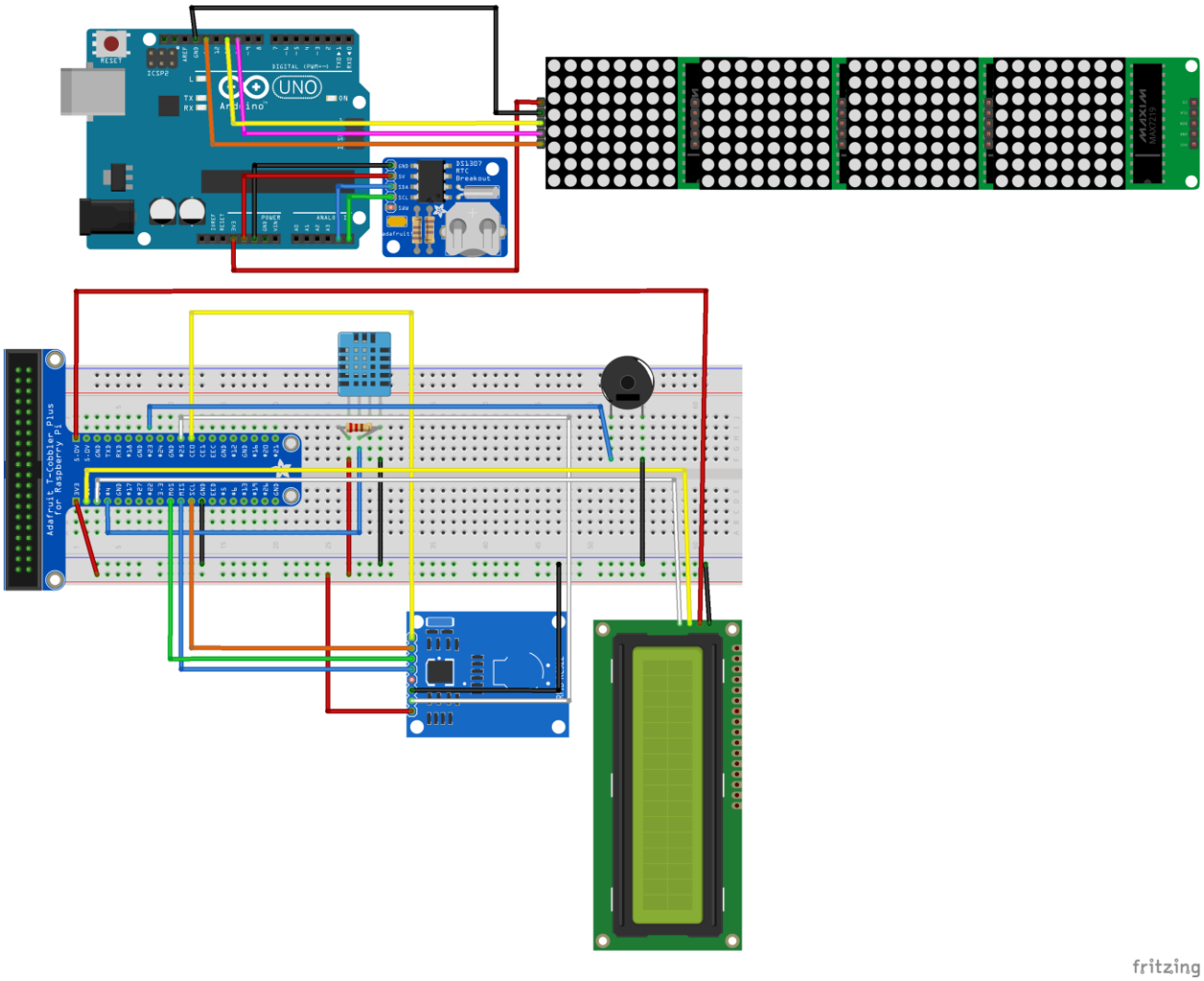


Figure 9 Full Look

Section 3
Software Requirements

Software checklist

If your applications needs the user to install additional Python or other libraries, pleasse provide here. A simple one like this is sufficient.

1. Python development libraries
2. SPI Python libraries
3. MFRC522-python library
4. rpi-lcd library
5. AWS Python library
6. Python library for AWS on your Raspberry Pi
7. Wire Library (Arduino IDE)
8. LedControl Library (Arduino IDE)
9. LEDMatrixDriver (Arduino IDE)

Software setup instructions

LEDControl for 4 in 1 MAX7219 Dot Matrix Display
https://github.com/wayoda/LedControl/releases
LEDMatrixController
https://github.com/stechiez/Arduino/tree/master/lib
DS1307 font library
https://github.com/stechiez/Arduino/tree/master/digitalClock
Extract the content of the zip files inside your [Arduino-->libraries] folder.

Wire Library is preloaded into the Arduino.

Install the Python development libraries
sudo apt-get install python-dev

Set up the SPI Python libraries since the card reader uses the SPI interface
cd ~
git clone https://github.com/lthiery/SPI-Py.git
cd ~/SPI-Py

sudo python setup.py install

Clone the MFRC522-python library to your home folder as follows:-
cd ~
git clone https://github.com/pimylifeup/MFRC522-python.git
cd ~/MFRC522-python
sudo python setup.py install

Install the rpi-lcd library using the commands below.
sudo pip install rpi-lcd

Install the Mosquitto broker and clients on your Raspberry Pi with this command
sudo apt-get install mosquitto mosquitto-clients

Install the latest AWS Python library with these commands
sudo pip install --upgrade --force-reinstall pip==9.0.3
sudo pip install AWSIoTPythonSDK --upgrade --disable-pip-version-check

Type the following command to install the AWS Command-Line Interface Client on your Raspberry Pi
sudo pip install awscli

If you already have awscli installed and want to upgrade to the latest version, you can type this instead
sudo pip install awscli --upgrade

Type the following command to install Boto, the Python library for AWS on your Raspberry Pi
sudo pip install botocore

If you already have boto installed and want to upgrade to the latest version, you can type this instead
sudo pip install botocore --upgrade
sudo pip install boto3 –upgrade

To set your AWS credentials which you will need for Botocore
Go to https://www.awseducate.com/signin/SiteLogin
Login > My Classrooms > Go to Classroom

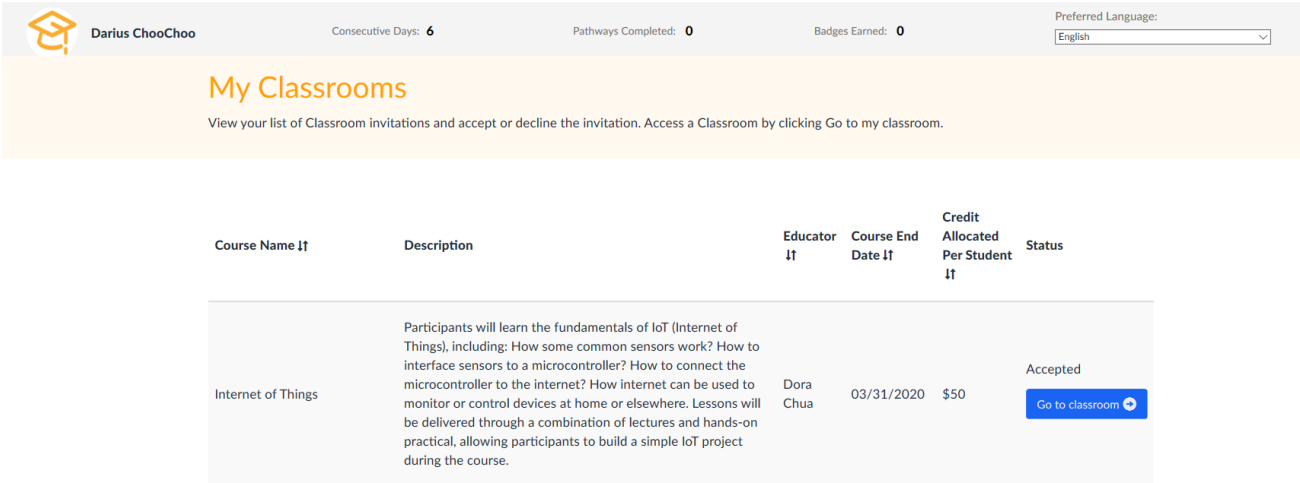


Figure 10 AWS

Continue > Account Details > Show

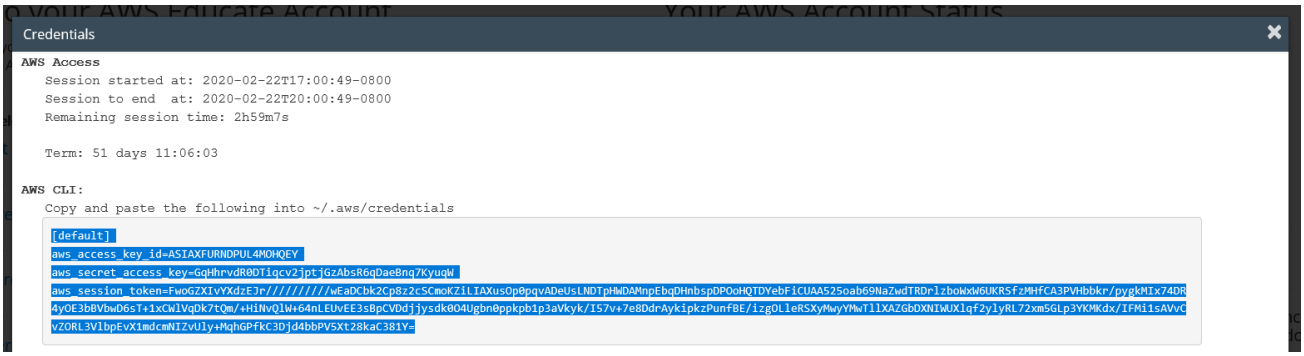


Figure 11 AWS Credentials

Copy highlighted portion

Sudo rm ~/.aws/credentials
Sudo nano ~/.aws/credentials

Paste what you copied from before
Ctrl 0 + Ctrl X to save

Section 4
Source codes

RFIDWorks.py

All source codes, including Python, HTML files etc

```
# Import SDK packages
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from time import sleep
from gpiozero import MCP3008, LED, Buzzer
from rpi_lcd import LCD
from picamera import PiCamera
import boto3
import botocore
import datetime as datetime
import RPi.GPIO as GPIO
import telepot
import sys
import Adafruit_DHT
import MFRC522
import signal
import json
import time

my_bot_token = '863095732:AAEehVDL5E-UeKPJw-jRsuuUuZhLTx0fRMA'
pin = 4
led = LED(18)
lcd = LCD()
ledOK = LED(24)
bz = Buzzer(23)
s3 = boto3.resource('s3')
adc = MCP3008(channel=0)
uid = None
prev_uid = None
continue_reading = True
full_path = '/home/pi/Desktop/image1.jpg'
file_name = 'image1.jpg'
bucket = 'sp-p1828881-s3-bucket'
exists = True
camera = PiCamera()
isATSOn = True

print("Program is running")
```

```
# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
    global continue_reading
    print "Ctrl+C captured, ending read."
    continue_reading = False
    GPIO.cleanup()

try:
    s3.meta.client.head_bucket(Bucket=bucket)
except botocore.exceptions.ClientError as e:
    error_code = int(e.response['Error']['Code'])
    if error_code == 404:
        exists = False

if exists == False:
  s3.create_bucket(Bucket=bucket,CreateBucketConfiguration={
    'LocationConstraint': 'us-east-1'})

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
    global continue_reading
    print "Ctrl+C captured, ending read."
    continue_reading = False
    GPIO.cleanup()

def takePhotoWithPiCam():
    camera.capture(full_path)

def onATS():
    global isATSOn
    isATSOn = True
    return "Got it, ATS is now turned on"

def offATS():
    global isATSOn
    isATSOn = False
    return "Got it, ATS is now turned off"

# Hook the SIGINT
signal.signal(signal.SIGINT, end_read)
print("Signal hooked")

# Create an object of the class MFRC522
mfrc522 = MFRC522.MFRC522()
print("mfrc522 created")
```

```
# Custom MQTT message callback
def customCallback(client, userdata, message):
print("Received a new message: ")
print(message.payload)
print("from topic: ")
print(message.topic)
print("--------------\n\n")

def respondToMsg(msg):
    chat_id = msg['chat']['id']
    command = msg['text']
    print('Got command:{}'.format(command))
    if command == 'offATS':
        bot.sendMessage(chat_id, offATS())
    if command == 'onATS':
        bot.sendMessage(chat_id, onATS())

host = "amitv187sde3m-ats.iot.us-east-1.amazonaws.com"
rootCAPath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"
print("certificates initialized")

bot = telepot.Bot(my_bot_token)
bot.message_loop(respondToMsg)
print('Listening for RPI Commands')

my_rpi = AWSIoTMQTTClient("PubSub-1828881")
my_rpi.configureEndpoint(host, 8883)
my_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)

my_rpi.configureOfflinePublishQueueing(-1)  # Infinite offline Publish queueing
my_rpi.configureDrainingFrequency(2)  # Draining: 2 Hz
my_rpi.configureConnectDisconnectTimeout(10)  # 10 sec
my_rpi.configureMQTTOperationTimeout(5)  # 5 sec
print("MQTT Configured")

# Connect and subscribe to AWS IoT
my_rpi.connect()
print("rpi connected")

my_rpi.subscribe("sensors/light", 1, customCallback)
print("rpi subcribed")
sleep(2)
```

```
# Publish to the same topic in a loop forever
loopCount = 0
while True:
    while isATSOn == True:
        # Init LEDs
        print (isATSOn)
        led.on()
        ledOK.off()
        bz.off()

        # Scan for cards
        (status,TagType) = mfrc522.MFRC522_Request(mfrc522.PICC_REQIDL)

        print("Scanning for cards")
        lcd.text('Scanning', 1)
        loopCount = loopCount+1


        # If a card is found
        if status == mfrc522.MI_OK:

            print("status ok")
            ledOK.on()
            bz.on()
            led.off()
            bz.off()
            lcd.text('Welcome', 1)

            # Get the UID of the card
            (status,uid) = mfrc522.MFRC522_Anticoll()
            message = ("New card detected! UID of card is {}".format(uid))
            message2 = {}
            message2["deviceid"]="deviceid_dariuschoo"

            now = datetime.datetime.now()
            message2["datetimeid"] = now.isoformat()
            message2["value"] = message
            #message2[]

            humidity, temperature = Adafruit_DHT.read_retry(11, pin)
            message2["temperature"] = temperature
            print(temperature)
```

```
        my_rpi.publish("sensors/light", json.dumps(message2), 1)

        takePhotoWithPiCam()
        s3.Object(bucket, file_name).put(Body=open(full_path, 'rb'))
        print("File uploaded")

        sleep(1)

    while isATSOn == False:
        # Init LEDs
        led.on()
        ledOK.off()
        lcd.text('No ATS', 1)
        sleep(1)
```

digitalClock.ino

```
#include "LEDMatrixDriver.hpp"
#include <Wire.h>
#include "RtcDS1307.h"


RtcDS1307<TwoWire> Rtc(Wire);
const uint8_t LEDMATRIX_CS_PIN = 10;

const int LEDMATRIX_SEGMENTS = 4;
const int LEDMATRIX_WIDTH    = LEDMATRIX_SEGMENTS * 8;

LEDMatrixDriver lmd(LEDMATRIX_SEGMENTS, LEDMATRIX_CS_PIN);

const int ANIM_DELAY = 60;

void setup() {
     Serial.begin(57600);

    Serial.print("compiled: ");
    Serial.print(__DATE__);
    Serial.println(__TIME__);

        Rtc.Begin();

    RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
    printDateTime(compiled);
    Serial.println();

    if (!Rtc.IsDateTimeValid())
```

```
    {
      if (Rtc.LastError() != 0)
      {
        Serial.print("RTC communications error = ");
        Serial.println(Rtc.LastError());
      }
      else
      {
        Serial.println("RTC lost confidence in the DateTime!");
        Rtc.SetDateTime(compiled);
      }
    }

    if (!Rtc.GetIsRunning())
    {
      Serial.println("RTC was not actively running, starting now");
      Rtc.SetIsRunning(true);
    }

    RtcDateTime now = Rtc.GetDateTime();
    if (now < compiled)
    {
      Serial.println("RTC is older than compile time!  (Updating DateTime)");
      Rtc.SetDateTime(compiled);
    }
    else if (now > compiled)
    {
      Serial.println("RTC is newer than compile time. (this is expected)");
    }
    else if (now == compiled)
    {
      Serial.println("RTC is the same as compile time! (not expected but all is fine)");
    }

    // never assume the Rtc was last configured by you, so
    // just clear them to your needed state
    Rtc.SetSquareWavePin(DS1307SquareWaveOut_Low);



  lmd.setEnabled(true);
  lmd.setIntensity(2);   // 0 = low, 10 = high
}

int x=0, y=0;   // start top left
```

```
const byte MAX7219_Dot_Matrix_font [95] [8]= {
{0,0,0,0,0,0,0,0}, // SPACE
{0x10,0x18,0x18,0x18,0x18,0x00,0x18,0x18}, // EXCL
{0x28,0x28,0x08,0x00,0x00,0x00,0x00,0x00}, // QUOT
{0x00,0x0a,0x7f,0x14,0x28,0xfe,0x50,0x00}, // #
{0x10,0x38,0x54,0x70,0x1c,0x54,0x38,0x10}, // $
{0x00,0x60,0x66,0x08,0x10,0x66,0x06,0x00}, // %
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, // &
{0x00,0x10,0x18,0x18,0x08,0x00,0x00,0x00}, // '
{0x02,0x04,0x08,0x08,0x08,0x08,0x08,0x04}, // (
{0x40,0x20,0x10,0x10,0x10,0x10,0x10,0x20}, // )
{0x00,0x10,0x54,0x38,0x10,0x38,0x54,0x10}, // *
{0x00,0x08,0x08,0x08,0x7f,0x08,0x08,0x08}, // +
{0x00,0x00,0x00,0x00,0x00,0x18,0x18,0x08}, // COMMA
{0x00,0x00,0x00,0x00,0x7e,0x00,0x00,0x00}, // -
{0x00,0x00,0x00,0x00,0x00,0x00,0x06,0x06}, // DOT
{0x00,0x04,0x04,0x08,0x10,0x20,0x40,0x40}, // /
{0x00,0x38,0x44,0x4c,0x54,0x64,0x44,0x38}, // 0
{0x04,0x0c,0x14,0x24,0x04,0x04,0x04,0x04}, // 1
{0x00,0x30,0x48,0x04,0x04,0x38,0x40,0x7c}, // 2
{0x00,0x38,0x04,0x04,0x18,0x04,0x44,0x38}, // 3
{0x00,0x04,0x0c,0x14,0x24,0x7e,0x04,0x04}, // 4
{0x00,0x7c,0x40,0x40,0x78,0x04,0x04,0x38}, // 5
{0x00,0x38,0x40,0x40,0x78,0x44,0x44,0x38}, // 6
{0x00,0x7c,0x04,0x04,0x08,0x08,0x10,0x10}, // 7
{0x00,0x3c,0x44,0x44,0x38,0x44,0x44,0x78}, // 8
{0x00,0x38,0x44,0x44,0x3c,0x04,0x04,0x78}, // 9
{0x00,0x18,0x18,0x00,0x00,0x18,0x18,0x00}, // :
{0x00,0x18,0x18,0x00,0x00,0x18,0x18,0x08}, // ;
{0x00,0x10,0x20,0x40,0x80,0x40,0x20,0x10}, // <
{0x00,0x00,0x7e,0x00,0x00,0xfc,0x00,0x00}, // =
{0x00,0x08,0x04,0x02,0x01,0x02,0x04,0x08}, // >
{0x00,0x38,0x44,0x04,0x08,0x10,0x00,0x10}, // ?
{0x00,0x30,0x48,0xba,0xba,0x84,0x78,0x00}, // @
{0x00,0x1c,0x22,0x42,0x42,0x7e,0x42,0x42}, // A
{0x00,0x78,0x44,0x44,0x78,0x44,0x44,0x7c}, // B
{0x00,0x3c,0x44,0x40,0x40,0x40,0x44,0x7c}, // C
{0x00,0x7c,0x42,0x42,0x42,0x42,0x44,0x78}, // D
{0x00,0x78,0x40,0x40,0x70,0x40,0x40,0x7c}, // E
{0x00,0x7c,0x40,0x40,0x78,0x40,0x40,0x40}, // F
{0x00,0x3c,0x40,0x40,0x5c,0x44,0x44,0x78}, // G
{0x00,0x42,0x42,0x42,0x7e,0x42,0x42,0x42}, // H
{0x00,0x7c,0x10,0x10,0x10,0x10,0x10,0x7e}, // I
{0x00,0x7e,0x02,0x02,0x02,0x02,0x04,0x38}, // J
```

```
{0x00,0x44,0x48,0x50,0x60,0x50,0x48,0x44}, // K
{0x00,0x40,0x40,0x40,0x40,0x40,0x40,0x7c}, // L
{0x00,0x82,0xc6,0xaa,0x92,0x82,0x82,0x82}, // M
{0x00,0x42,0x42,0x62,0x52,0x4a,0x46,0x42}, // N
{0x00,0x3c,0x42,0x42,0x42,0x42,0x44,0x38}, // O
{0x00,0x78,0x44,0x44,0x48,0x70,0x40,0x40}, // P
{0x00,0x3c,0x42,0x42,0x52,0x4a,0x44,0x3a}, // Q
{0x00,0x78,0x44,0x44,0x78,0x50,0x48,0x44}, // R
{0x00,0x38,0x40,0x40,0x38,0x04,0x04,0x78}, // S
{0x00,0x7e,0x90,0x10,0x10,0x10,0x10,0x10}, // T
{0x00,0x42,0x42,0x42,0x42,0x42,0x42,0x3e}, // U
{0x00,0x42,0x42,0x42,0x42,0x44,0x28,0x10}, // V
{0x80,0x82,0x82,0x92,0x92,0x92,0x94,0x78}, // W
{0x00,0x42,0x42,0x24,0x18,0x24,0x42,0x42}, // X
{0x00,0x44,0x44,0x28,0x10,0x10,0x10,0x10}, // Y
{0x00,0x7c,0x04,0x08,0x7c,0x20,0x40,0xfe}, // Z
{ 0x00, 0x7F, 0x7F, 0x41, 0x41, 0x00, 0x00, 0x00 }, // '['
{ 0x01, 0x03, 0x06, 0x0C, 0x18, 0x30, 0x60, 0x00 }, // backslash
{ 0x00, 0x41, 0x41, 0x7F, 0x7F, 0x00, 0x00, 0x00 }, // ']'
{ 0x08, 0x0C, 0x06, 0x03, 0x06, 0x0C, 0x08, 0x00 }, // '^'
{ 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80 }, // '_'
{ 0x00, 0x00, 0x03, 0x07, 0x04, 0x00, 0x00, 0x00 }, // '`'
{0x00,0x70,0x08,0x08,0x78,0x48,0x48,0x7c}, // 'a'
{0x00,0x40,0x40,0x40,0x78,0x44,0x44,0x78},//b
{0x00,0x3c,0x40,0x40,0x40,0x40,0x3c,0x00},//c
{0x00,0x04,0x04,0x04,0x3c,0x44,0x44,0x3c},//d
{0x00,0x38,0x44,0x64,0x5c,0x40,0x42,0x3c},//e
{0x00,0x18,0x20,0x20,0x70,0x20,0x20,0x20},//f
{0x00,0x70,0x48,0x48,0x38,0x08,0x48,0x38},//g
{0x00,0x40,0x40,0x40,0x70,0x50,0x50,0x50},//h
{0x00,0x10,0x00,0x10,0x10,0x10,0x10,0x10},//i
{0x00,0x10,0x00,0x10,0x10,0x10,0x50,0x30},//j
{0x00,0x40,0x40,0x48,0x50,0x60,0x50,0x48},//k
{0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x68},//l
{0x00,0x00,0x40,0x36,0x2a,0x2a,0x2a,0x2a},//m
{0x00,0x00,0x40,0x3c,0x24,0x24,0x24,0x24},//n
{0x00,0x00,0x38,0x44,0x44,0x44,0x38,0x00},//o
{0x00,0xb0,0x48,0x48,0x70,0x40,0x40,0x40},//p
{0x00,0x32,0x4c,0x4c,0x34,0x04,0x04,0x04},//q
{0x00,0x08,0x50,0x30,0x20,0x20,0x20,0x20},//r
{0x00,0x38,0x44,0x40,0x38,0x04,0x44,0x38},//s
{0x00,0x20,0x20,0x70,0x20,0x20,0x20,0x18},//t
{0x00,0x00,0x48,0x48,0x48,0x48,0x3c,0x00},//u
{0x00,0x00,0x44,0x44,0x44,0x44,0x28,0x10},//v
{0x00,0x00,0x00,0x44,0x44,0x54,0x28,0x00},//w
```

```
{0x00,0x00,0x42,0x24,0x18,0x18,0x24,0x42},//x
{0x00,0x48,0x28,0x18,0x08,0x08,0x28,0x18},//y
{0x00,0x00,0x78,0x10,0x20,0x40,0x78,0x00},
{ 0x00, 0x00, 0x00, 0x77, 0x77, 0x00, 0x00, 0x00 }, // '|'
{ 0x41, 0x41, 0x77, 0x3E, 0x08, 0x08, 0x00, 0x00 }, // '}'
{ 0x02, 0x03, 0x01, 0x03, 0x02, 0x03, 0x01, 0x00 }, // '~'
};  //  end of MAX7219_Dot_Matrix_font

RtcDateTime time_to_update;
int count=0;

void loop()
{
  count++;
  Serial.println(count);
  if(count > 20)
  {
    count = 0;
    RtcDateTime now = Rtc.GetDateTime();
    time_to_update = now;
  }
  printDateTime(time_to_update);
}

void drawString(const char* text, int len, int x, int y )
{
  for( int idx = 0; idx < len; idx ++ )
  {
    char c = text[idx];// - 32;
    if((c >= ' ') && (c <= 126))
    {
      c = c - ' ';
        if( x + idx * 8  > LEDMATRIX_WIDTH )
      return;

    if( 8 + x + idx * 8 > 0 )
    {
      drawLetter( MAX7219_Dot_Matrix_font[c], x + idx * 8, y, 8, 8 );
    }
    }
  }
}

void drawLetter(  const byte* Letter, int x, int y, int width, int height )
{
```

```
  byte mask = B10000000;

  for( int iy = 0; iy < height; iy++ )
  {
    for( int ix = 0; ix < width; ix++ )
    {
      lmd.setPixel(x + ix, y + iy, (bool)(Letter[iy] & mask ));
      mask = mask >> 1;
    }
    mask = B10000000;
  }
}

#define countof(a) (sizeof(a) / sizeof(a[0]))

void printDateTime(const RtcDateTime& dt)
{
    char datestring[20];

    snprintf_P(datestring,
           countof(datestring),
          PSTR("%02u:%02u:%02u"),
        //  dt.Month(),
        //dt.Day(),
          //dt.Year(),
           dt.Hour(),
           dt.Minute(),
           dt.Second() );
            int len = strlen(datestring);
  drawString(datestring, len, x, 0);
Serial.print(dt.Hour());
Serial.print(dt.Minute());
  lmd.display();
  if( --x < len * -8 ) {
    x = LEDMATRIX_WIDTH;
  }
  delay(ANIM_DELAY);
}
```

LEDMatrixDriver.hpp

```
/*
* LEDMatrixDriver.h
*
*  Created on: 30.03.2017
```

```
*     Author: Bartosz Bielawski Credits: embpic, edited: Edward 20/02/2020
*/


#ifndef LEDMATRIXDRIVER_H_
#define LEDMATRIXDRIVER_H_

#include <SPI.h>

#ifdef ESP32
#include <cstring>
#endif

#ifdef USE_ADAFRUIT_GFX
#include <Adafruit_GFX.h>
class LEDMatrixDriver: public Adafruit_GFX
#else
class LEDMatrixDriver
#endif
{
//commands as defined in the datasheet
const static uint16_t ENABLE =          0x0C00;
const static uint16_t TEST =        0x0F00;
const static uint16_t INTENSITY =   0x0A00;
const static uint16_t SCAN_LIMIT =  0x0B00;
const static uint16_t DECODE =          0x0900;


public:
const static uint8_t INVERT_SEGMENT_X = 1;
const static uint8_t INVERT_DISPLAY_X = 2;
const static uint8_t INVERT_Y = 4;

//with N segments and ssPin as SS,
//flags describe segment orientation (optional)
//an already allocated buffer can be provided as well (optional)
LEDMatrixDriver(uint8_t N, uint8_t ssPin, uint8_t flags = 0, uint8_t* frameBuffer = nullptr);
#ifdef USE_ADAFRUIT_GFX
virtual
#endif
~LEDMatrixDriver();

//we don't want to copy the object
LEDMatrixDriver(const LEDMatrixDriver& other) = delete;
LEDMatrixDriver(LEDMatrixDriver&& other) = delete;
```

```
LEDMatrixDriver& operator=(const LEDMatrixDriver& other) = delete;

#ifdef USE_ADAFRUIT_GFX
virtual void writePixel(int16_t x, int16_t y, uint16_t color) {setPixel(x,y,color);}
virtual void drawPixel(int16_t x, int16_t y, uint16_t color) {setPixel(x,y,color);}
virtual void endWrite(void) {if (not manualDisplayRefresh) display();}
void setManualDisplayRefresh(bool enabled) {manualDisplayRefresh = enabled;}
#endif

//all these commands work on ALL segments
void setEnabled(bool enabled);
//display brightness: 0 - 15
void setIntensity(uint8_t level);
void setPixel(int16_t x, int16_t y, bool enabled);
bool getPixel(int16_t x, int16_t y) const;
//sets pixels in the column acording to value (LSB => y=0)
void setColumn(int16_t x, uint8_t value);
uint8_t getSegments() const {return N;}

uint8_t* getFrameBuffer() const {return frameBuffer;}

//functions for 7-segment displays
//number of digits displayed (0 -> 1 digit, 7 -> 8 digits)
void setScanLimit(uint8_t level);
void setDecode(uint8_t mask);
void setDigit(uint16_t digit, uint8_t value, bool dot = false);

//flush the data to the display
void display();
//flush a single row to the display
void displayRow(uint8_t row) {_displayRow(row);}
//clear the framebuffer
void clear() {memset(frameBuffer, 0, 8*N);}

enum class scrollDirection
{
scrollUp = 0,
scrollDown,
scrollLeft,
scrollRight
};

//scroll the framebuffer 1 pixel in the given direction
void scroll( scrollDirection direction );
```

```
// BCD Code B values
const static uint8_t BCD_DASH =     0x0A;
const static uint8_t BCD_E =        0x0B;
const static uint8_t BCD_H =        0x0C;
const static uint8_t BCD_L =        0x0D;
const static uint8_t BCD_P =        0x0E;
const static uint8_t BCD_BLANK =    0x0F;

private:
uint8_t* _getBufferPtr(int16_t x, int16_t y) const;
void _sendCommand(uint16_t command);
void _displayRow(uint8_t row);

const uint8_t N;
SPISettings spiSettings;
uint8_t flags;
uint8_t* frameBuffer;
bool selfAllocated;
uint8_t ssPin;

#ifdef USE_ADAFRUIT_GFX
bool manualDisplayRefresh = true;
#endif
};

#endif /* LEDMATRIXDRIVER_H_ */
```

RtcDS1307.h

```
#ifndef __RTCDS1307_H__
#define __RTCDS1307_H__

#include <Arduino.h>
#include "RtcDateTime.h"
#include "RtcUtility.h"

//I2C Slave Address
const uint8_t DS1307_ADDRESS = 0x68;

//DS1307 Register Addresses
const uint8_t DS1307_REG_TIMEDATE   = 0x00;
const uint8_t DS1307_REG_STATUS     = 0x00;
const uint8_t DS1307_REG_CONTROL    = 0x07;
const uint8_t DS1307_REG_RAMSTART   = 0x08;
```

```
const uint8_t DS1307_REG_RAMEND     = 0x3f;
const uint8_t DS1307_REG_RAMSIZE = DS1307_REG_RAMEND - DS1307_REG_RAMSTART;

//DS1307 Register Data Size if not just 1
const uint8_t DS1307_REG_TIMEDATE_SIZE = 7;

// DS1307 Control Register Bits
const uint8_t DS1307_RS0   = 0;
const uint8_t DS1307_RS1   = 1;
const uint8_t DS1307_SQWE  = 4;
const uint8_t DS1307_OUT   = 7;

// DS1307 Status Register Bits
const uint8_t DS1307_CH      = 7;

enum DS1307SquareWaveOut
{
   DS1307SquareWaveOut_1Hz  =  0b00010000,
   DS1307SquareWaveOut_4kHz =  0b00010001,
   DS1307SquareWaveOut_8kHz =  0b00010010,
   DS1307SquareWaveOut_32kHz = 0b00010011,
   DS1307SquareWaveOut_High =  0b10000000,
   DS1307SquareWaveOut_Low =   0b00000000,
};

template<class T_WIRE_METHOD> class RtcDS1307
{
public:
   RtcDS1307(T_WIRE_METHOD& wire) :
      _wire(wire),
      _lastError(0)
   {
   }

   void Begin()
   {
      _wire.begin();
   }

   uint8_t LastError()
   {
      return _lastError;
   }

   bool IsDateTimeValid()
```

```
{
   return GetIsRunning();
}

bool GetIsRunning()
{
   uint8_t sreg = getReg(DS1307_REG_STATUS);
   return !(sreg & _BV(DS1307_CH));
}

void SetIsRunning(bool isRunning)
{
   uint8_t sreg = getReg(DS1307_REG_STATUS);
   if (isRunning)
   {
      sreg &= ~_BV(DS1307_CH);
   }
   else
   {
      sreg |= _BV(DS1307_CH);
   }
   setReg(DS1307_REG_STATUS, sreg);
}

void SetDateTime(const RtcDateTime& dt)
{
   // retain running state
   uint8_t sreg = getReg(DS1307_REG_STATUS) & _BV(DS1307_CH);

   // set the date time
   _wire.beginTransmission(DS1307_ADDRESS);
   _wire.write(DS1307_REG_TIMEDATE);

   _wire.write(Uint8ToBcd(dt.Second()) | sreg);
   _wire.write(Uint8ToBcd(dt.Minute()));
   _wire.write(Uint8ToBcd(dt.Hour())); // 24 hour mode only

   // RTC Hardware Day of Week is 1-7, 1 = Monday
   // convert our Day of Week to Rtc Day of Week
   uint8_t rtcDow = RtcDateTime::ConvertDowToRtc(dt.DayOfWeek());

   _wire.write(Uint8ToBcd(rtcDow));
   _wire.write(Uint8ToBcd(dt.Day()));
   _wire.write(Uint8ToBcd(dt.Month()));
   _wire.write(Uint8ToBcd(dt.Year() - 2000));
```

```
    _lastError = _wire.endTransmission();
}

RtcDateTime GetDateTime()
{
    _wire.beginTransmission(DS1307_ADDRESS);
    _wire.write(DS1307_REG_TIMEDATE);
    _lastError = _wire.endTransmission();
    if (_lastError != 0)
    {
        RtcDateTime(0);
    }

    _wire.requestFrom(DS1307_ADDRESS, DS1307_REG_TIMEDATE_SIZE);
    uint8_t second = BcdToUint8(_wire.read() & 0x7F);
    uint8_t minute = BcdToUint8(_wire.read());
    uint8_t hour = BcdToBin24Hour(_wire.read());

    _wire.read();  // throwing away day of week as we calculate it

    uint8_t dayOfMonth = BcdToUint8(_wire.read());
    uint8_t month = BcdToUint8(_wire.read());
    uint16_t year = BcdToUint8(_wire.read()) + 2000;

    return RtcDateTime(year, month, dayOfMonth, hour, minute, second);
}

void SetMemory(uint8_t memoryAddress, uint8_t value)
{
    uint8_t address = memoryAddress + DS1307_REG_RAMSTART;
    if (address <= DS1307_REG_RAMEND)
    {
        setReg(address, value);
    }
}

uint8_t GetMemory(uint8_t memoryAddress)
{
    uint8_t value = 0;
    uint8_t address = memoryAddress + DS1307_REG_RAMSTART;
    if (address <= DS1307_REG_RAMEND)
    {
        value = getReg(address);
    }
```

```
    return value;
}

uint8_t SetMemory(uint8_t memoryAddress, const uint8_t* pValue, uint8_t countBytes)
{
    uint8_t address = memoryAddress + DS1307_REG_RAMSTART;
    uint8_t countWritten = 0;
    if (address <= DS1307_REG_RAMEND)
    {
        _wire.beginTransmission(DS1307_ADDRESS);
        _wire.write(address);

        while (countBytes > 0 && address <= DS1307_REG_RAMEND)
        {
            _wire.write(*pValue++);
            address++;
            countBytes--;
            countWritten++;
        }

        _lastError = _wire.endTransmission();
    }
    return countWritten;
}

uint8_t GetMemory(uint8_t memoryAddress, uint8_t* pValue, uint8_t countBytes)
{
    uint8_t address = memoryAddress + DS1307_REG_RAMSTART;
    uint8_t countRead = 0;
    if (address <= DS1307_REG_RAMEND)
    {
        if (countBytes > DS1307_REG_RAMSIZE)
        {
            countBytes = DS1307_REG_RAMSIZE;
        }

        _wire.beginTransmission(DS1307_ADDRESS);
        _wire.write(address);
        _lastError = _wire.endTransmission();
        if (_lastError != 0)
        {
            return 0;
        }

        countRead = _wire.requestFrom(DS1307_ADDRESS, countBytes);
```

```
        countBytes = countRead;

        while (countBytes-- > 0)
        {
            *pValue++ = _wire.read();
        }
    }

    return countRead;
  }

  void SetSquareWavePin(DS1307SquareWaveOut pinMode)
  {
    setReg(DS1307_REG_CONTROL, pinMode);
  }

private:
  T_WIRE_METHOD& _wire;
  uint8_t _lastError;

  uint8_t getReg(uint8_t regAddress)
  {
    _wire.beginTransmission(DS1307_ADDRESS);
    _wire.write(regAddress);
    _lastError = _wire.endTransmission();
    if (_lastError != 0)
    {
      return 0;
    }

    // control register
    _wire.requestFrom(DS1307_ADDRESS, (uint8_t)1);

    uint8_t regValue = _wire.read();
    return regValue;
  }

  void setReg(uint8_t regAddress, uint8_t regValue)
  {
    _wire.beginTransmission(DS1307_ADDRESS);
    _wire.write(regAddress);
    _wire.write(regValue);
    _lastError = _wire.endTransmission();
  }
};
```

```
#endif // __RTCDS1307_H__
```

server.py

```
SAMPLE - Copy and paste your codes here
from flask import Flask, render_template, jsonify,request

app = Flask(__name__)

import dynamodb as db
import jsonconverter as jsonc

@app.route("/api/getdata",methods=['POST','GET'])
def apidata_getdata():
    print("ENTERED GET DATA")
    if request.method == 'POST' or request.method == 'GET':
        try:
            data = {'chart_data': jsonc.data_to_json(db.get_data_from_dynamodb()),
             'title': "IOT Data"}
            return jsonify(data)

        except:
            import sys
            print(sys.exc_info()[0])
            print(sys.exc_info()[1])

@app.route("/")
def home():
    return render_template("index.html")


app.run(debug=True,host="0.0.0.0")
```

index.html

```
SAMPLE - Copy and paste your codes here
<!doctype html>
<head>
    <style> #chartDiv {width:100%;}</style>
    <title>Google Charts with Flask</title>
    <script type="text/javascript" src="https://code.jquery.com/jquery-3.2.1.js"></script>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
        google.charts.load('current', {'packages':['corechart','table']});
```

```
// Set a callback to run when the Google Visualization API is loaded.
google.charts.setOnLoadCallback(googlecharts_is_ready);

var chart;
var graphdata;

function reset_status_messages(){
   $("#status").html("")
}

function googlecharts_is_ready(){
   $("#buttonloadchart").show()
   $("#buttonloadchart").click()
   $("#status").html("Google charts is ready")
}

function loadChart(){
     getData_and_drawChart()
}

function getData_and_drawChart(){
   getNewData()
}


function getNewData(){
   $("#status").html("Fetching data to plot graph...");

   jQuery.ajax({
      url: "/api/getdata" ,
      type: 'POST',
      error: function(jqXHR,textStatus, errorThrown ){
          console.log("Error while ajax:" + textStatus)
      },
      success: function(ndata, textStatus, xhr){
         console.log("INDEX HTML GETNEWDATA: "+ndata)


         //console.log(ndata.chart_data)
         $("#status").html("Data fetched! Now plotting graph!");
         chartdata = ndata.chart_data
         graphdata = createDataTable(chartdata)
         drawLineChart(graphdata)
         drawDataTable(graphdata)
         $("#status").html("Graph plotted");
```

```
            }//end success
        });//end ajax
      } //end getNewData

    function createDataTable(newdata){
        graphdata = new google.visualization.DataTable();
        graphdata.addColumn('string', 'Time');
        graphdata.addColumn('number', 'Temperature');
        var newdata = JSON.parse(newdata);

        for (index=0;index<newdata.length;index++){
            console.log("INDEX HTML GET TEMPERATURE: "+parseInt(newdata[index].temperature))
            datetime = (newdata[index].datetimeid)
            datetime = datetime.substring(0, 19) //+ "+0000"
            jsdatetime = new Date(Date.parse(datetime));
            jstime = jsdatetime.toLocaleTimeString();
            light = parseInt(newdata[index].temperature);
            graphdata.addRows([[jstime,light]]);
        }//end for
        return graphdata
    }

    function drawDataTable(graphdata){
        var table = new google.visualization.Table(document.getElementById('table_div'));
        table.draw(graphdata, {showRowNumber: true, width: '100%', height: '100%'});

    }//end drawTable

    function drawLineChart(graphdata) {
        chart = new google.visualization.LineChart(
        document.getElementById('chart_div'));
        chart.draw(graphdata, {legend: 'none', vAxis: {baseline: 0},
            colors: ['#A0D100']});
        return
    } //end drawChart

    $(document).ready(function(){
        reset_status_messages()

        setInterval(function () {
            loadChart()
        }, 3000);
    });

</script>
```

```
</head>
<body>
    <input id="buttonloadchart" type="button" onclick="loadChart()" value="Update graph">
    <div id="status"></div>
    <div id="chart_div" style="width:100%"></div>
    <div id="table_div" style="width:100%"></div>

</body>
```

jsonconverter.py

```
from decimal import Decimal
import json
import datetime
import numpy

class GenericEncoder(json.JSONEncoder):

  def default(self, obj):
    if isinstance(obj, numpy.generic):
      return numpy.asscalar(obj)
    elif isinstance(obj, Decimal):
      return str(obj)
    elif isinstance(obj, datetime.datetime):
      return obj.strftime('%Y-%m-%d %H:%M:%S')
    elif isinstance(obj, Decimal):
      return float(obj)
    else:
      return json.JSONEncoder.default(self, obj)

def data_to_json(data):
  json_data = json.dumps(data,cls=GenericEncoder)
  print("JSONCONVERTER JSON DATA: "+json_data)
  return json_data
```

Dynamodb.py

```
def get_data_from_dynamodb():
  print("ENTERED GET DATA FROM DYNAMODB")
  try:
      import boto3
      from boto3.dynamodb.conditions import Key, Attr

      dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
```

```
        table = dynamodb.Table('iotca2')

        print("DYNAMODB TABLE COUNT: "+str(table.item_count))
        startdate = '2020'

        response = table.query(
            KeyConditionExpression=Key('deviceid').eq('deviceid_dariuschoo')
                            & Key('datetimeid').begins_with(startdate),
            ScanIndexForward=False
        )

        items = response['Items']

        n=10 # limit to last 10 items
        data = items[:n]
        data_reversed = data[::-1]

        return data_reversed

    except:
        import sys
        print(sys.exc_info()[0])
        print(sys.exc_info()[1])


if __name__ == "__main__":
    get_data_from_dynamodb()
```

Section 5
Task List

A table listing members names and the parts of the assignment they worked on

Name of member	Part of project worked on	Contribution percentage
Gary	LCD Display	33%
Darius	RFID, Camera, Buzzer, DynamoDB, S3, Telegram, Web Server	33%
Edward	LED Matrix,Clock Module	33%

Section 6
References

References to online materials used

https://www.instructables.com/id/User-Manual-MAX7219-Dot-Matrix-4-in-1/
https://www.instructables.com/id/Digital-Clock-Using-Arduino-and-Led-Dot-Matrix-Dis/

Github Link: https://github.com/EdwardBrick/IOTCA2byIDK.git

-- End of CA2 Step-by-step tutorial --