CIS 4930 011

Internet-of-Things (IoT) System Design

Final Project Report

|  |  |
| --- | --- |
| Today’s Date: | 12/08/20 |
| Your Name: | Md Mahamudul Islam |
| Team: | 1. Name (U#) 29218485   Project Contribution: |
| No. of Hours Spent: | Almost 48 Hrs |
| Exercise Difficulty:  (Easy, Average, Hard) | Hard |
| Any Other Feedback: | No |

* 1. **Project 1 (50 pts)** ([The Nerd](https://create.arduino.cc/projecthub/arduino/the-nerd-0144f9))

1. (5 pts) In your own words, describe the project.

A small assistant robot which engages your attention to show that you are not alone, you are crowded with electrical machines.

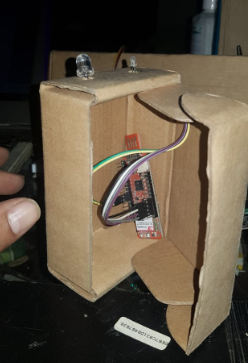
It has a battery saving mode for stop wasting battery. The Nerd will wake up each half hour to scan the networks around it. If it detects new networks, it will store them and go back to sleep in low power mode

It also tells you current time and date if it is connected to internet

1. (7.5 pts) A picture of **your** board setup.



1. (7.5 pts) A picture of **your** system setup.



1. (15 pts) Your sketch code.

#include <SPI.h>

#include <WiFi101.h>

#include <FlashStorage.h>

#include <RTCZero.h>

#include <WiFiUdp.h>

#include "ArduinoLowPower.h"

WiFiUDP udp;

WiFiUDP Udp;

RTCZero rtc;

#define MAGIC\_NUMBER 0x7423 // arbitrary number to double check the validity of SSID

#define MaxNet 30 // max amount of network to be saved

// RGB LED pins

int redPin = 6;

int greenPin = 8;

int bluePin = 7;

const char\* home\_ssid = SECRET\_SSID; // your network SSID (name)

const char\* password = SECRET\_PSWD; // your network password

int BuzzerPin = 9;

int SensorPin = A2;

int PosToBeSaved = 0; // Variable used to navigate the array of saved networks

int daily\_amount\_of\_food = 12; // The amount of food per day needed to survive

int sleeping\_time = 1800000; // 30 min \*60 sec \*1000 millisec

bool atHome = false;

bool hungry=true;

bool justWokeUp=false;

// Struct of variable to be saved in flash memory

typedef struct {

int magic;

boolean valid[MaxNet];

char SSIDs[MaxNet][100];

int alive\_days;

int last\_time\_feeded;

} Networks;

FlashStorage(my\_flash\_store, Networks);

Networks values;

void setup() {

Serial.begin(115200);

delay(2000);

pinMode(redPin, OUTPUT);

pinMode(greenPin, OUTPUT);

pinMode(bluePin, OUTPUT);

rtc.begin(); // enable real time clock functionalities

// set up a WakeUp function, it will be triggered each time the Nerd exit the sleeping mode

LowPower.attachInterruptWakeup(RTC\_ALARM\_WAKEUP, WakeUp, CHANGE);

values = my\_flash\_store.read(); // Read values from flash memory

if (values.magic == MAGIC\_NUMBER) { // If token is correct print saved networks

Serial.println("saved data:");

Serial.println("");

for (int a = 0; a < MaxNet; a++) {

if (values.valid[a]) {

Serial.println(values.SSIDs[a]);

} else {

PosToBeSaved = a;

}

}

}

}

void loop() {

int SensorValue=analogRead(SensorPin);

// Awaking notification

if(SensorValue>30 && justWokeUp){

justWokeUp=false;

setColor(0, 255, 0); // green

tone(BuzzerPin, 31, 200); // tone(Pin, Note, Duration);

delay(200);

setColor(0, 0, 0); // off

noTone(BuzzerPin);

delay(1000);

}

// Check if the Nerd has been fed within 2 days

if(rtc.getEpoch() - values.last\_time\_feeded >= 86400\*2){ // 86400 is the number of seconds in one day

// DIE :(

SOS();

values.alive\_days = 0 ;

values.last\_time\_feeded = rtc.getEpoch();

empty\_network\_array();

}

// End of the day, empty the network array and go to sleep

if(rtc.getHours() == 23 && rtc.getMinutes() >= 30){

// Empty the array of network

values.alive\_days +=1;

empty\_network\_array();

hungry=true;

LowPower.sleep(3600\*8); // sleep 8 hours

}

if(!atHome) check\_home();

if (!atHome && hungry) {

Serial.println("checking for network");

// Temporarly save the number of networks

int networks\_already\_saved = PosToBeSaved;

getNetwork();

// compare the two values and complain if no new network is detected

if (networks\_already\_saved == PosToBeSaved) SOS();

if(PosToBeSaved >= daily\_amount\_of\_food) hungry=false; // check if had enough food

if(SensorValue < 30) LowPower.sleep(sleeping\_time); // snooze if dark

}

else if(atHome && !hungry) {

Serial.println("back to sleep");

LowPower.sleep(sleeping\_time);

}

else if(atHome && hungry){

if(SensorValue < 30) LowPower.sleep(sleeping\_time); // back to sleep if it's dark

SOS();

}

// Set color status feedback

if(PosToBeSaved >= 8){ // if starving show red

setColor(255, 0, 0); // Red

}

else if(PosToBeSaved > 4 && PosToBeSaved < 8){

setColor(255, 255, 0); // yellow

}

else{

setColor(0, 255, 0); // green

}

}

void WakeUp() {

Serial.println("awake");

atHome = false;

justWokeUp =true;

}

void check\_home() {

int numSsid = WiFi.scanNetworks();

if (numSsid != -1) {

for (int thisNet = 0; thisNet < numSsid; thisNet++) {

delay(100);

if (strncmp(WiFi.SSID(thisNet), home\_ssid, 100) == 0) {

Serial.println("Yay, I'm home \n");

atHome = true;

connect\_WiFi();

}

}

}

}

void connect\_WiFi() {

if (WiFi.status() != WL\_CONNECTED) {

while (WiFi.begin(home\_ssid, password) != WL\_CONNECTED) {

delay(500);

}

Serial.println("WiFi connected \n");

GetCurrentTime();

printTime();

}

}

// Feed the Nerd with networks's SSID

void getNetwork() {

// scan for nearby networks:

Serial.println("\*Scan Networks\*");

int numSsid = WiFi.scanNetworks();

delay(1000);

if (numSsid == -1)

{

Serial.println("There are no WiFi networks here..");

} else {

Serial.print("number of available networks: ");

Serial.println(numSsid);

// print the network number and name for each network found:

for (int thisNet = 0; thisNet < numSsid; thisNet++) {

Serial.print("SSID: ");

Serial.println(WiFi.SSID(thisNet));

delay(500);

char\* net = WiFi.SSID(thisNet);

bool canBeSaved = true;

// check if the network has already been saved

for (int a = 0; a < PosToBeSaved ; a++) {

if (values.valid[a]) {

if (strncmp(net, values.SSIDs[a], 100) == 0 || strnlen(net, 100) == 0) {

Serial.println("Not saved");

canBeSaved = false;

}

}

}

// Store ssid name

if (canBeSaved && PosToBeSaved < MaxNet) {

if (strlen(net) + 1 < 100 && strlen(net) > 0) { // check if the SSID name fits 100 bytes

memset(values.SSIDs[PosToBeSaved], 0, sizeof(values.SSIDs[PosToBeSaved])); // set all characters to zero

memcpy(values.SSIDs[PosToBeSaved], net, strlen(net) + 1); // copy "net" to values.SSDs[thisNet]

values.valid[PosToBeSaved] = true;

values.last\_time\_feeded = rtc.getEpoch();

values.magic = MAGIC\_NUMBER;

my\_flash\_store.write(values);

Serial.println(String(values.SSIDs[PosToBeSaved]) + " saved in position " + String(PosToBeSaved));

PosToBeSaved ++;

}

else {

Serial.println(" network skipped");

}

}

}

}

}

// Reset the array in which networks are saved

void empty\_network\_array(){

for(int a = 0; a < PosToBeSaved; a++ ){

values.valid[a] = false;

}

values.magic = 0;

my\_flash\_store.write(values);

}

void SOS(){

for(int a = 0; a< 3; a++){

setColor(255, 0, 0); // Red

tone(BuzzerPin, 31, 100); // tone(Pin, Note, Duration);

delay(100);

setColor(0, 0, 0); // off

noTone(BuzzerPin);

delay(50);

}

delay(1000);

for(int a = 0; a< 3; a++){

setColor(255, 0, 0); // Red

tone(BuzzerPin, 31, 2000); // tone(Pin, Note, Duration);

delay(1000);

}

for(int a = 0; a< 3; a++){

setColor(255, 0, 0); // Red

tone(BuzzerPin, 31, 100); // tone(Pin, Note, Duration);

delay(100);

setColor(0, 0, 0); // off

noTone(BuzzerPin);

delay(50);

}

delay(10000);

}

// Send RGB values to the LED pins

void setColor(int red, int green, int blue){

analogWrite(redPin, red);

analogWrite(greenPin, green);

analogWrite(bluePin, blue);

}

1. (15 pts) A short video of your functioning IoT System.

In the Zip file

**Project 2 (50 pts): (**[Puzzle Box](https://create.arduino.cc/projecthub/arduino/puzzlebox-c1f374))

1. (5 pts) In your own words, describe the project.

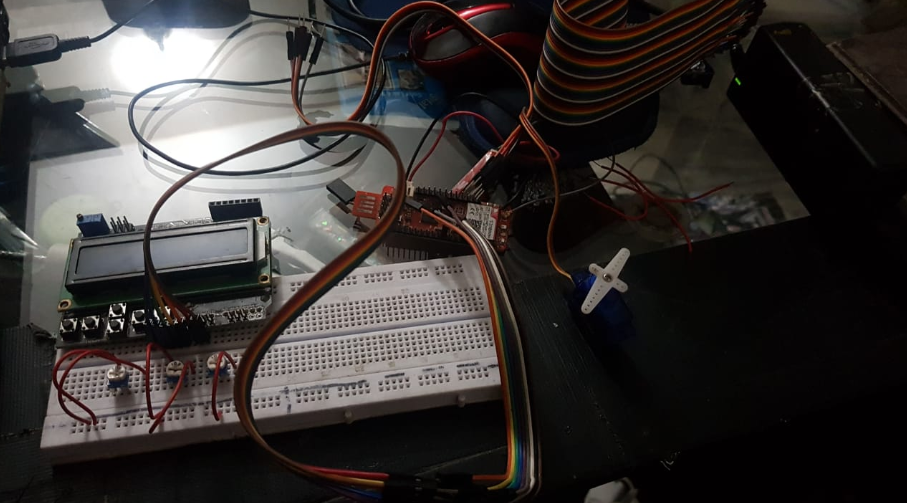
In this project we are going to use 5 main things

1. Servo motor
2. Lcd screen
3. Blynk application
4. Potentiometer
5. RGB led

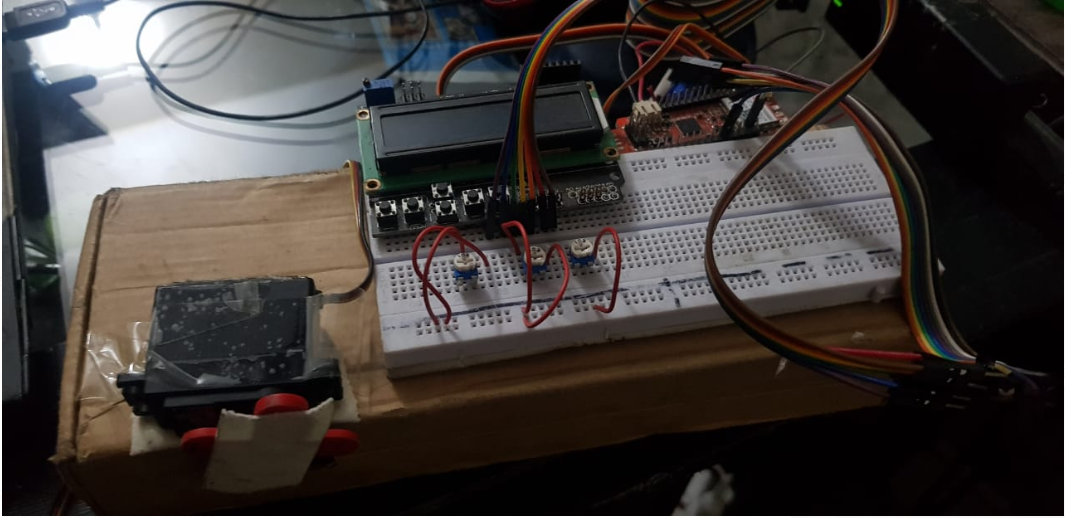
This is a simple puzzle box which will only open if a specific combination is achieved it is a type of locker which no one else can open without knowing the combination of potentiometer.

Blynk application is a new thing for us because we did not use it before.

1. (7.5 pts) A picture of **your** board setup.



1. (7.5 pts) A picture of **your** system setup.



1. (15 pts) Your sketch code.

#include <LiquidCrystal.h>

#include <SPI.h>

#include <WiFi101.h>

#include <BlynkSimpleWiFiShield101.h>

#include <Servo.h>

#define buzzerPin 1

#include "Melody.h"

// RGB LED pins

int redPin = 6;

int greenPin = 8;

int bluePin = 7;

const char\* ssid = SECRET\_SSID; // your network SSID (name)

const char\* password = SECRET\_PSWD; // your network password

char auth[] = SECRET\_TOKEN; // your Blynk API token

// LCD screen pins

const int rs = 12,

en = 11,

d4 = 2,

d5 = 3,

d6 = 4,

d7 = 5;

bool start = true;

// Variables to store the combination value

// Set the intitial combination to ( 1 1 1 )

int SliderValueOne = 1;

int SliderValueTwo = 1;

int SliderValueThree = 1;

int pos = 0; // variable to store the servo position

Servo myservo; // create servo object to control a servo

// Blynk functions to retrive values

BLYNK\_WRITE(V1) {

SliderValueOne = param.asInt(); // assigning incoming value from pin V1 to a variable

}

BLYNK\_WRITE(V2) {

SliderValueTwo = param.asInt(); // assigning incoming value from pin V1 to a variable

}

BLYNK\_WRITE(V3) {

SliderValueThree = param.asInt(); // assigning incoming value from pin V1 to a variable

}

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {

pinMode(redPin, OUTPUT);

pinMode(greenPin, OUTPUT);

pinMode(bluePin, OUTPUT);

pinMode(buzzerPin, OUTPUT);

analogWrite(A3, 0); // set the brightness of the LCD screen to the maximum value

Serial.begin(9600);

lcd.begin(16, 2); // begin LCD screen with 16 columns and 2 rows

Blynk.begin(auth, ssid, password); // start Blynk functionalities

myservo.attach(9); // attaches the servo on pin 9 to the servo object

myservo.write(pos); // set the servo in position 0

}

void loop() {

// Variambles to temporarily store the combination

int Temp\_Slider\_One\_value = SliderValueOne;

int Temp\_Slider\_Two\_value = SliderValueTwo;

int Temp\_Slider\_Three\_value = SliderValueThree;

Blynk.run(); // poll new combination values from the online app

// check if combination values are changed and print them on the console

if(Temp\_Slider\_One\_value != SliderValueOne || Temp\_Slider\_Two\_value != SliderValueTwo || Temp\_Slider\_Three\_value != SliderValueThree){

Serial.print("New combination: ");

Serial.print(SliderValueOne);

Serial.print(" ");

Serial.print(SliderValueTwo);

Serial.print(" ");

Serial.println(SliderValueThree);

}

int PotOne = map(analogRead(A0), 100, 1023, 0, 9);

int PotTwo = map(analogRead(A1), 100, 1023, 0, 9);

int PotThree = map(analogRead(A2), 100, 1023, 0, 9);

lcd.setCursor(0, 0);

lcd.print(PotOne);

lcd.setCursor(2, 0);

lcd.print(PotTwo);

lcd.setCursor(4, 0);

lcd.print(PotThree);

if (start) {

giveColorFeedback(PotOne, PotTwo, PotThree);

if (PotOne == SliderValueOne && PotTwo == SliderValueTwo && PotThree == SliderValueThree) {

play\_jingle();

open\_the\_box();

blinkGreenLed();

start = false;

}

}

if(!start) {

if(PotOne == 0 && PotTwo == 0 && PotThree == 0){

close\_the\_box();

start = true;

}

}

}

// Give feedback based on how close the potentiometer are to the combination value

// The more it's close the warmer is the color of the LED

void giveColorFeedback(int PotOne, int PotTwo, int PotThree) {

if (abs(PotOne - SliderValueOne) <= 1 && abs(PotTwo - SliderValueTwo) <= 1 && abs(PotThree - SliderValueThree) <= 1 ) {

// Red

setColor(255, 0, 0);

}

else if (abs(PotOne - SliderValueOne) <= 3 && abs(PotTwo - SliderValueTwo) <= 3 && abs(PotThree - SliderValueThree) <= 3 ) {

// yellow

setColor(255, 255, 0);

}

else if (abs(PotOne - SliderValueOne) <= 4 && abs(PotTwo - SliderValueTwo) <= 4 && abs(PotThree - SliderValueThree) <= 4 ) {

// aqua

setColor(0, 255, 255);

}

else {

// blue

setColor(0, 0, 255);

}

}

void blinkGreenLed() {

for (int a = 0; a < 2; a++) {

for (int b = 0; b <= 255; b += 5) {

setColor(0, b, 0);

delay(5);

}

for (int b = 255; b >= 0; b -= 5) {

setColor(0, b, 0);

delay(5);

}

}

for (int b = 0; b <= 255; b += 5) {

setColor(0, b, 0);

delay(5);

}

}

// Send RGB values to the LED pins

void setColor(int red, int green, int blue){

analogWrite(redPin, red);

analogWrite(greenPin, green);

analogWrite(bluePin, blue);

}

void open\_the\_box(){

for (pos = 0; pos <= 90; pos += 1) { // goes from 0 degrees to 90 degrees

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

}

void close\_the\_box(){

for (pos = 90; pos >= 0; pos -= 1) { // goes from 90 degrees to 0 degrees

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

}

1. (15 pts) A short video of your functioning IoT System.

In the zip file