SMART HOME

1. INTRODUCTION

1.1 Objective and Scope of The Work

In this project work I am implementing an smart Home controlling system based on ThingSpeak IoT Cloud Platform and Temperature sensor, LDR(light dependent resister)/ photo sensor, PIR sensor and push button. ThingSpeak is an open source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP protocol over the Internet. ThingSpeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates.

The Smart home control system needs no manual operation for switching ON/ OFF the light, when there is need of light it was controlled by ThingSpeak IoT Platform through the mobile app. When darkness rises to a certain configured value the ThingSpeak Platform will send an alert mesage, then we can turn ON the light through the mobile app. Along with LDR, PIR sensors the system is having one push button for manual operation, i.e. to turn on/off the light.

Below are the major tasks which are performed by the system :

1. Reading the digital input from ADC which is derived from Temperature, Light sensor and PIR sensors and upload these values to the ThingSpeak Channel.
2. Controlling the light turning On/Off based on ThingSpeak TalkBack commannds and button.

This proposed research work would be implemented using **Arduino** based embedded system design methodology, which includes Embedded hardware and firmware design modules. This project would be carried out with ATMEGA328P based Arduino board and Aurduino driven Embedded EDA Tool kits.

1.2 Hardware requirements:

- ATmega328 Microcontroller based Arduino UNO Board
- WiFi-101 Module
- LM35 Temperature Sensor
- LDR, PIR sensors
- Buzzer, Led, Push Button
- Bread Board, Connecting wires

1.3 Software requirements:

- Embedded C
- codebender Compiler
- fritzing.0.9.3b.64.pc
- ThingSpeak IoT Cloud Platform

2. HARDWARE IMPLEMENTATION OF DASH BOARD

2.1 Dash Board

The implementation of the Dash board can be divided in two parts.

1. Hardware implementation
2. Firmware implementation

The Hardware implementation deals in drawing the schematic on the plane paper according to the application, testing the schematic design over the breadboard using the various components.

The block diagram discusses about the required components of Dash board and working condition is explained using circuit diagram.

2.2 Block Diagram of UNO board

The block diagram of UNO board is as shown in Fig 2.1. It consists of microcontroller, WiFi-101 module, power supply unit, LM35, LDR and PIR sensors, Buzzer, Led and push button (switch)..

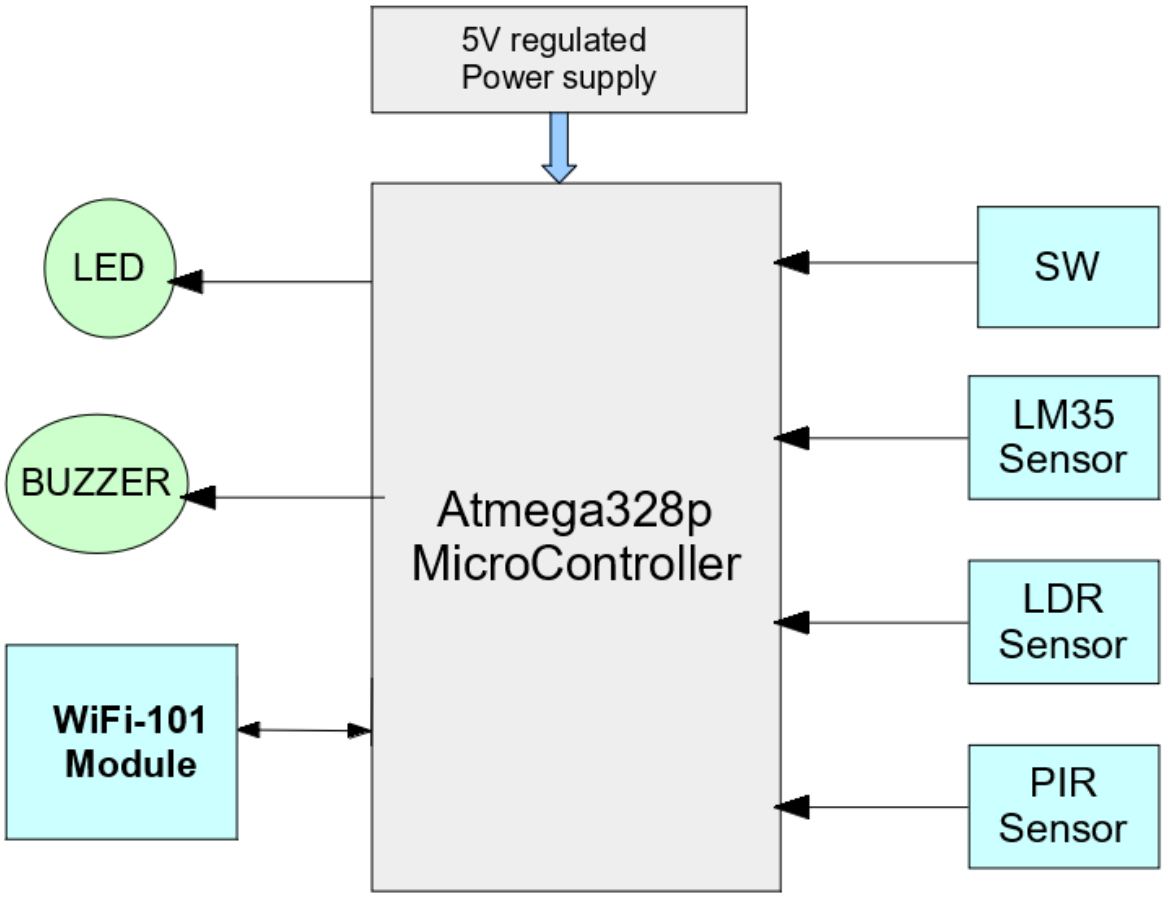


Fig.2.1 the Hardware diagram of the system

2.3 Circuit Diagram of Dash board :

I have used the fritzing.0.9.3b.64.pc software for Schematic design. Fig 2.2 shows the circuit diagram of the Dash board.

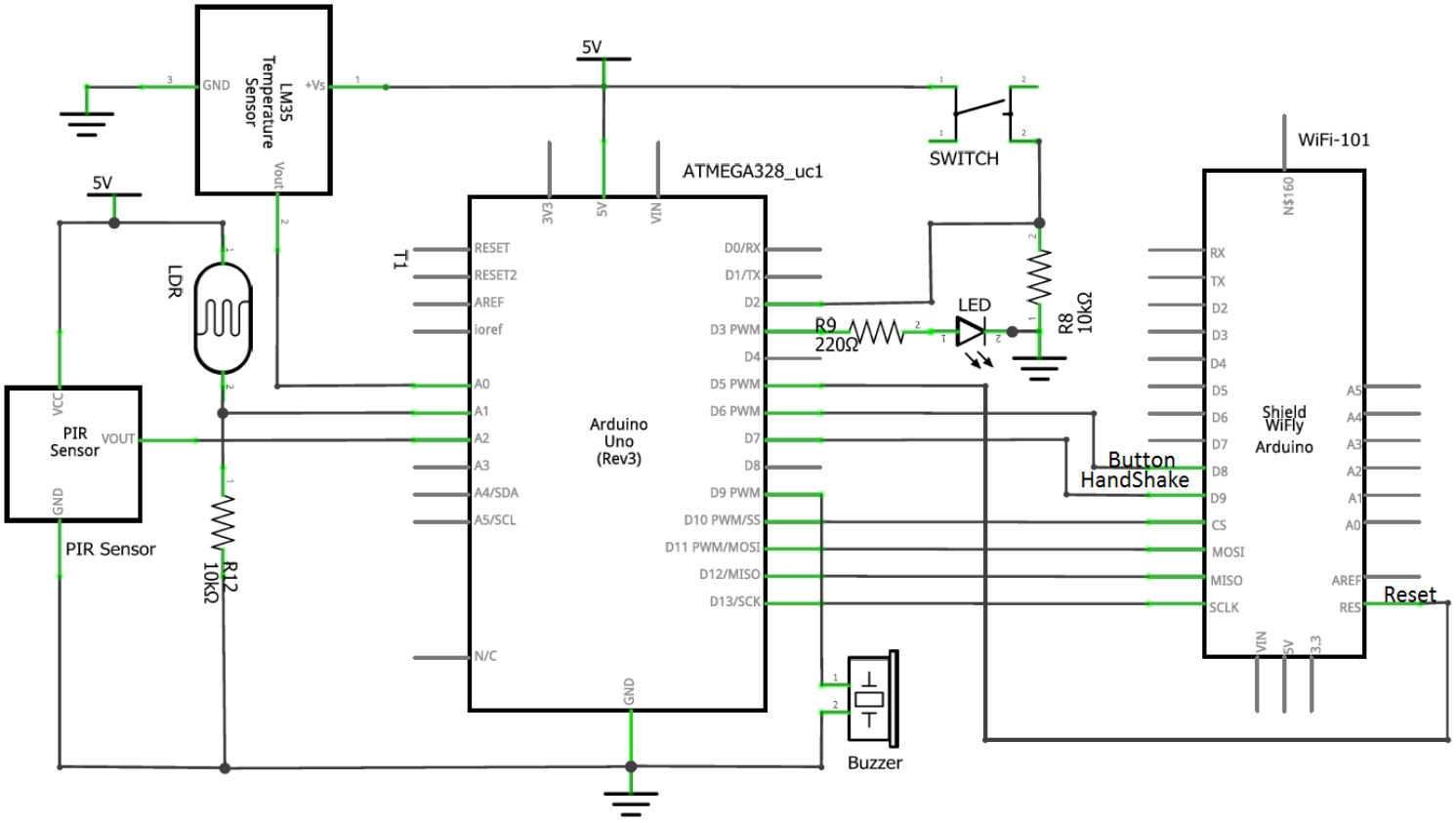


Fig 2.2: Circuit (Schematic) diagram of Dash board

2.4 Connections of Modules:

PORT-A:

A5	A4	A3	A2	A1	A0
			PIR	LDR	LM35

PORT-D :

13	12	11	10	9	8	7	6	5	4	3	2
WIFI SCK	WIFI MISO	WIFI MOSI	WIFI NSS	BUZZER	-	WIFI HANDSHAKE	WIFI BUTTON	WIFI RST	-	LED	BUTTON

Fig. 2.3: Port Map for Smart Home System

5. SOFTWARE IMPLEMENTATION OF DASH BOARD

I have used the Arduino IDE (cross-platform) for developing code for the system, which includes support for the **C** and C++ programming languages. I used Embedded C for project implementation. A program written with Arduino IDE is called a "sketch". A typical Arduino C/C++ sketch consist of two functions (setup (initializes settings), loop (called repeatedly in infinite loop)) that are compiled and linked with a program stub main() into an executable cyclic program.

5.1 Hardware Initialisation :

- Set PIN 2 as INPUT pin which is connected to push button named PIN_SWITCH and attached the buttonSwitchPushed ISR to the gpio pin 2.
- Set pin 3 and pin 4 are OUTPUT pins which are connected to the led and Buzzer respectively.
- Initialised the h/w serial port with 115200 baudrate for debugging.
- Calibrated the Temperature, LDR sensors and set the CutOff values for sensors.
- Initialised the WiFi-101 Module and connected to the Network.

5.2 The working of System :

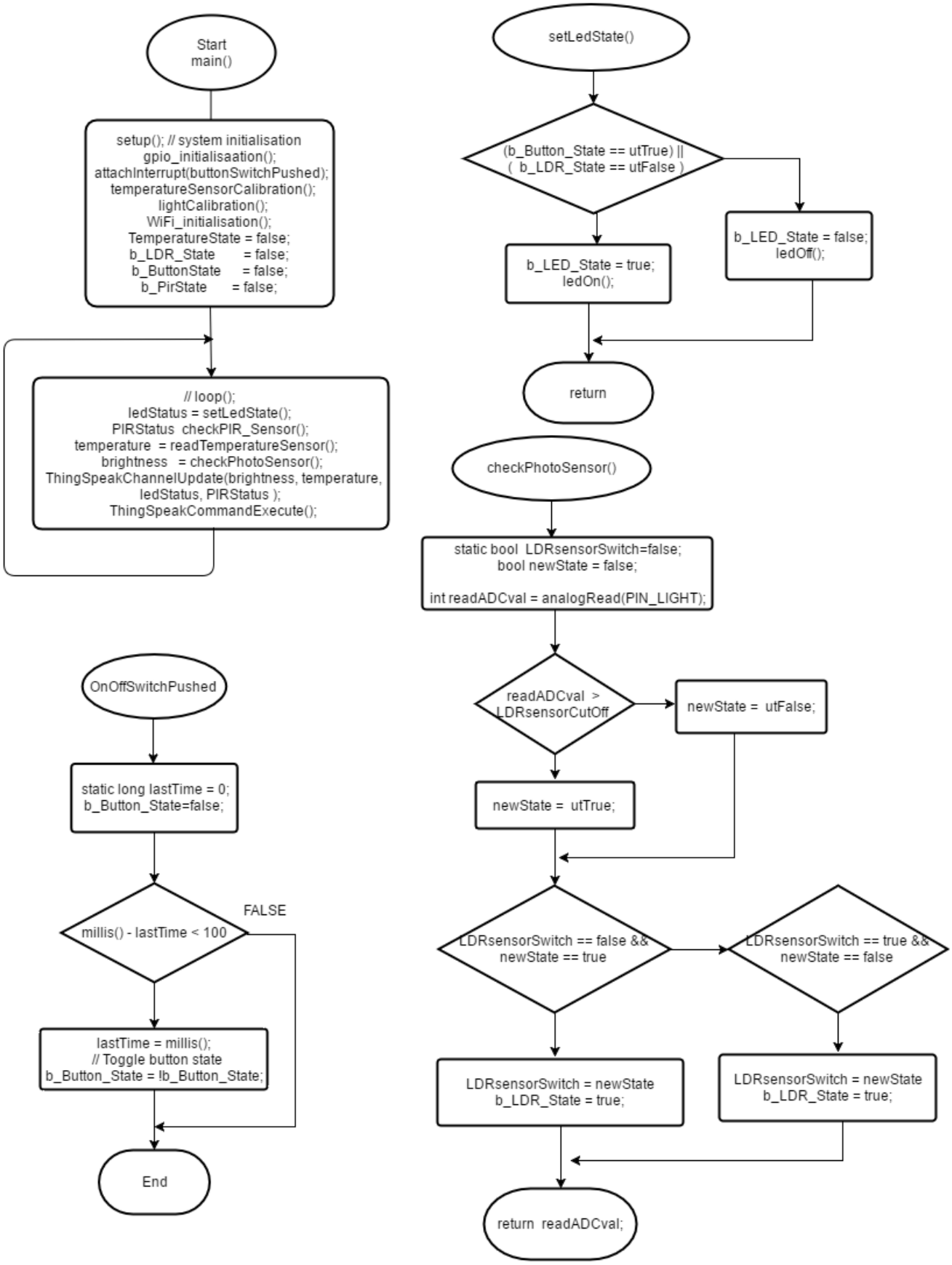
- ButtonSwitchPushed**()** : When the interrupt is triggred it toggles the b_ButtonState flag.
- setLedState**()** : It sets the LED on/off and *b_Led_State* flag based on b_ButtonState flag.
- CheckPIR_Sensor**()** : This function reads the ADC value if it is greater than the threshold sets the b_PirState flag.
- CheckPhotoSensor**()** : This function reads the ADC value if it is greater than the threshold (LDRsensorCutOff) sets the *b_LDR_State* flag and returns the sensor value.
- ReadTemperatureSensor**()** : This function reads the ADC value and converts it into the farenhite degrees and returns the temperature value.
- ThingSpeakChannelUpdate**()** : Uploads the sensor values, led and button states to the ThingSpeak IoT platform.
- ThingSpeakCommandExecute**()**: Checks whether the TalkBack commands exists in the ThingSpeak TalkBack command queue, if exists performs the corresponding action (i.e. LED on/off).

5.3 The ThingSpeak IoT Services:

- In this project I created ***Smart Home*** ThingSpeak public Channel with 5 fields (Brightness, Temperature, LED Status, Led Control and PIR Motion).
- I created 5 Thing reactions on React App against the high temperature, low light, person motion detection, light on and light off status.

1. HIGH_TEMP : It triggers the *HIGH_TEMP Alert* of ThingHTTP request.
2. LOW_LIGHT : It triggers the *LOW_LIGHT Alert* of ThingHTTP request.
3. PERSON_DETECT : It triggers the *PIR Alert Call* of ThingHTTP request.
4. LED_ON : It tweets the ***lights_ON in your Smart Home*** message when the light turned on.
5. LED_OFF : It tweets the ***lights_ON in your Smart Home*** message when the light turned off.

- I created 6 ThingHTTP services.
  1. HIGH_TEMP Alert : This service sends the high temperature alert message to the mobile.
  2. LOW_LIGHT Alert : This service sends the low light alert message to the mobile.
  3. PIR Alert Call : This service makes the alert call to mobile, when the PIR sensor detects any person/ object.
  4. SmartHomeControl : This is the TimeControl Service, this service sends one Welcome message to the mobile per day.
  5. The SmartHomeControl : It sends the Welcome text message to the mobile when the TimeControl service triggers it.
  6. The Tweet_LED_ON/Tweet_LED_OFF services allows us to social control the LED based on Twitter.
     - Tweet_LED_ON: It creates *LED_ON* TalkBack Command when the TweetControl responds to the word #***LIGHTS_ON*** triggered in the tweet message.
     - Tweet_LED_OFF: It creates *LED_OFF* TalkBack Command when the TweetControl responds to the word #***LIGHTS_OFF*** triggered in the tweet message.
- I had set up an TimeControl service (SmartHomeTimeControl), it triggers SmartHomeControl ThingHTTP servie once per Day.
- Compiled and uploaded the sketch to the Arduino UNO board.
- Now we can turn the LED on and off by using the button. Open the Serial Console to see the debug messages when we push the button to turn the LED on/off.
- We can turn the LED on and off by using the ThingSpeak IoT Platform through the mobile app.
- We can monitor the field charts, stastics and channel status updates in ThingSpeak IoT platform.
- We can monitor the TalkCommands which are created by the mobile app to control the LED.
- We can turn on/off the light using Twitter by Tweeting the specified trigger messages. Also we can get the tweets when the light gets on/off through the mobile app or by manual operation (button).


Start
main()
setup(); // system initialisation
gpio_initialisaation();
attachInterrupt(buttonSwitchPushed);
temperatureSensorCalibration();
lightCalibration();
WiFi_initialisation();
TemperatureState = false;
b_LDR_State = false;
b_ButtonState = false;
b_PirState = false;
// loop();
ledStatus = setLedState();
PIRStatus checkPIR_Sensor();
temperature = readTemperatureSensor();
brightness = checkPhotoSensor();
ThingSpeakChannelUpdate(brightness, temperature,
ledStatus, PIRStatus);
ThingSpeakCommandExecute();
setLedState()
(b_Button_State == utTrue) ||
(b_LDR_State == utFalse)
b_LED_State = true;
ledOn();
b_LED_State = false;
ledOff();
return
checkPhotoSensor()
static bool LDRsensorSwitch=false;
bool newState = false;
int readADCval = analogRead(PIN_LIGHT);
readADCval >
LDRsensorCutOff
newState = utFalse;
newState = utTrue;
LDRsensorSwitch == false &&
newState == true
LDRsensorSwitch == true &&
newState == false
LDRsensorSwitch = newState
b_LDR_State = true;
LDRsensorSwitch = newState
b_LDR_State = true;
return readADCval;
OnOffSwitchPushed
static long lastTime = 0;
b_Button_State=false;
millis() - lastTime < 100
FALSE
lastTime = millis();
// Toggle button state
b_Button_State = !b_Button_State;
End

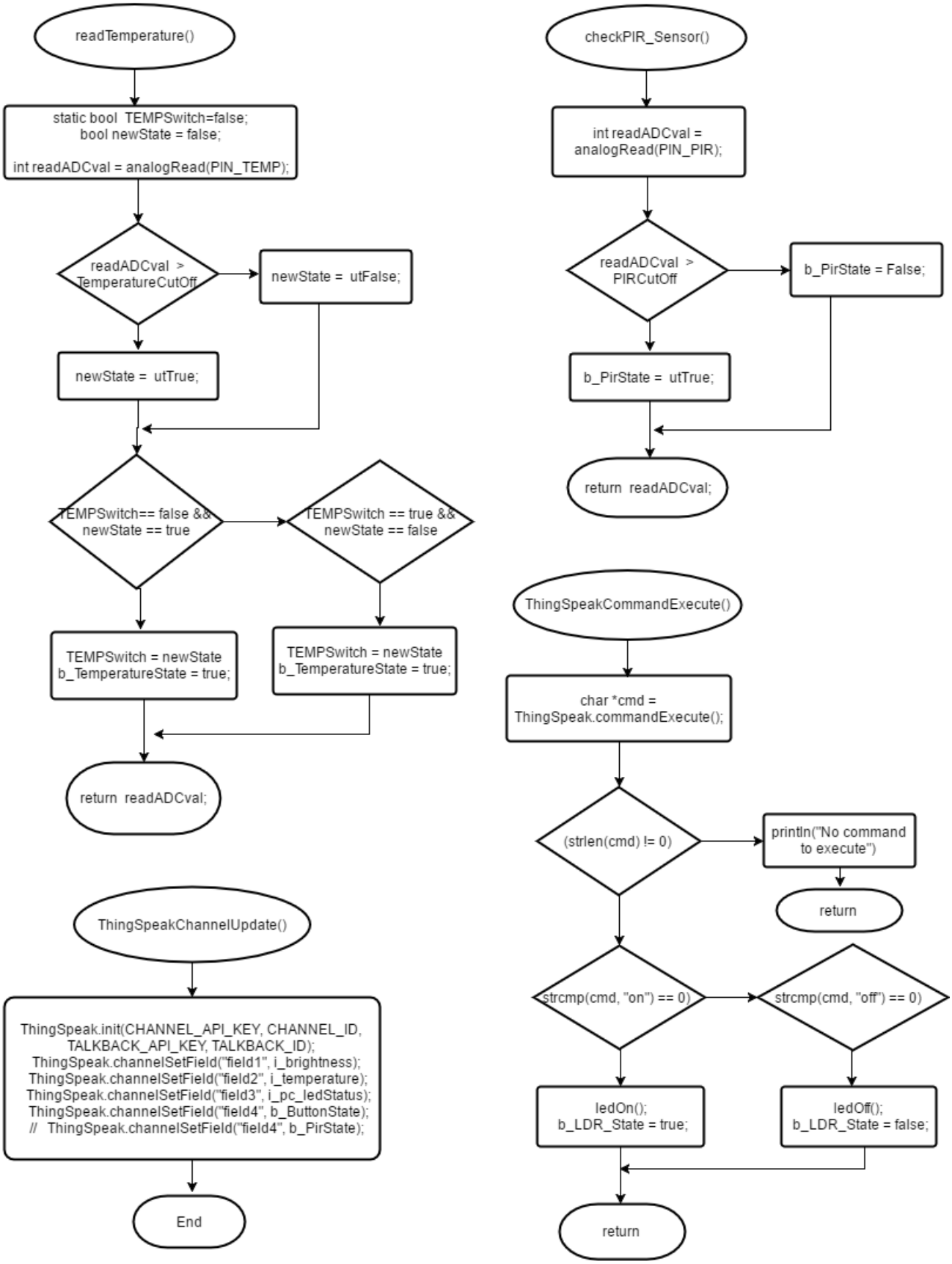
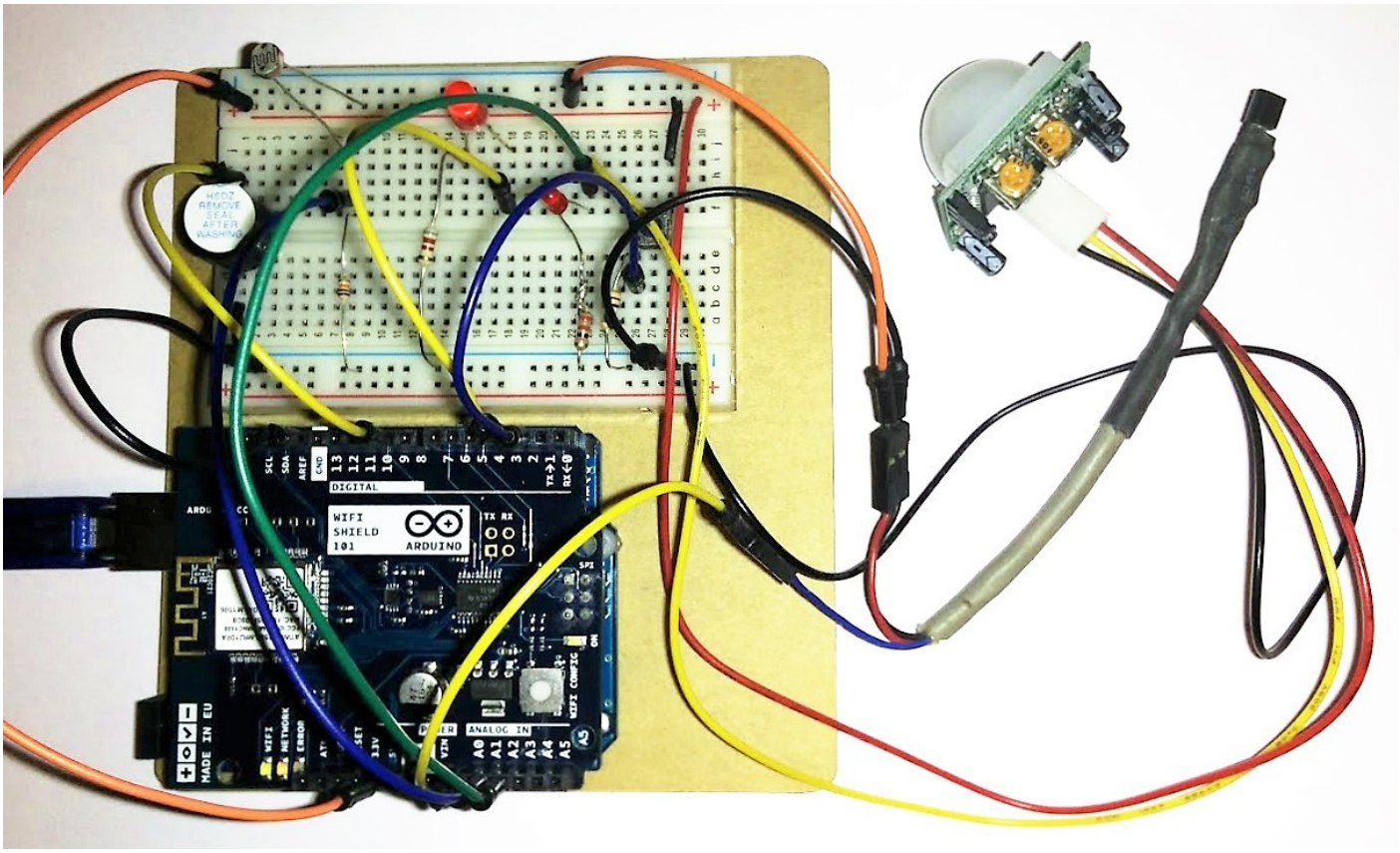


Fig. Flow Chart for Communication Unit

6. Appendix

Youtube video clip **:** https://www.youtube.com/watch?v=c4J2atbuOfA

Hardware setup : -



Serial Console Output of One Complete Session Run Which Demonstrates all features of the System:

Please vary temperature condition to start calibration

Temperature CutOff value is 671

Please vary light condition to start light calibration

Light cutoff value is 661

Free RAM: 627

Initializing WiFi Sheild 101 ...

Attempting to connect to SSID: NETGEAR30

SSID: NETGEAR30

IP Address: 192.168.0.16

signal strength (RSSI):-62 dBm

Fetch next command ...

Response code : 200

Response : 0

No command to execute

TEMPRATURE = 73.31*F 22.95*C

LDR sensor : 820

PIR sensor, Person detected : 394

Updating channel ...

&field1=820&field2=73.309&field3=on&field4=1&field5=1

Response code : 200

Response : 0

Fetch next command ...

Response code : 200

Response : on

Got command to execute: on

LED ON

Updating channel ...

&field1=820&field2=75.066&field3=on&field4=1&field5=0

Response code : 200

Response : 0

Fetch next command ...

Response code : 200

Response :

No command to execute

TEMPRATURE = 75.07*F 23.93*C

LDR sensor : 820

PIR sensor, Person Not detected : 1

Fetch next command ...

Response code : 200

Response : off

Got command to execute: off

LED OFF

Updating channel ...

&field1=820&field2=75.066&field3=off&field4=0&field5=0

Response code : 200

Response : 0