IoT Table Lamp using AWS IoT

C. Santhosh Sagar (RA1511004010157)
Electronics and Communication
SRM University, Kattankulathur

***Abstract*-- In this report the author shows the working and implementation of an Internet of things enabled table lamp. The table lamp works in two modes, firstly Bluetooth control mode, secondly motion detection mode. The sensor values from the PIR sensor are sent to Amazon Web Service's IoT web console where it can be further processed. Here the author sends the value to AWS Simple Notification Service which notifies the user if there is motion present via an email.**

***Keywords*-- AWS IoT, SNS, Simple Notification Service, PIR, Raspberry Pi, Arduino.**

I.INTRODUCTION

In this day and age we expect everything to be smart, fast and easily accessible and comfortable for us to use. We expect everything to understand our need s and desires and to work the way we want it to without much hassle. This project epitomizes how a humble thousand year old staple in every house the desk lamp can be made smart and be made a piece of tech that reflects the fast growth of wireless technology, connected devices and smart devices.

II.DESCRIPTION

The Smart Table lamp is based on Internet of things technology, Blueetooth and Motion Detection using PIR sensors.

The components we have used are:

- Arduino microcontroller
- 2-Channel Opt-coupler Relay
- HC-SR501 PIR Sensor
- HC-05 Bluetooth SPP module
- Raspberry Pi 3
- LM35 Temperature Sensor
- Incandescent Bulb
- 16x2 LCD Display
- Pushbutton
- 10k ohm Pot
- 1k Resistor
- Jumpers and Breadboard

III.MOTION DETECTION MODE

A.How PIR Sensors Work:

First let's explain the working principle. The module actually consists of a Pyroelectric sensor which generates energy when exposed to heat.

That means when a human or animal body will get in the range of the sensor it will detect a movement because the human or animal body emits heat energy in a form of infrared radiation. That's where the name of the sensor comes from, a Passive Infrared sensor. And the term "passive" means that sensor is not using any energy for detecting purposes, it just works by detecting the energy given off by the other objects.The module also consists a specially designed cover named Fresnel lens, which focuses the infrared signals onto the pyroelectric sensor.

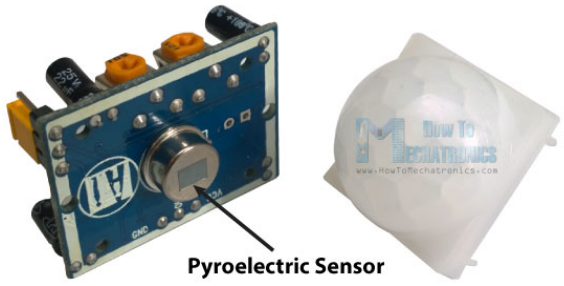

Figure 1. PIR Sensor

The module has just three pins, a Ground and a VCC for powering the module and an output pin which gives high logic level if an object is detected. Also it has two potentiometers. One for adjusting the sensitivity of the sensor and the other for adjusting the time the output signal stays high when object is detected. This time can be adjusted from 0.3 seconds up to 5 minutes.

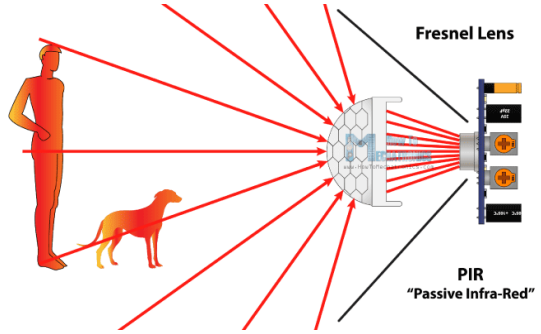

Figure 2. Detection of objects

The module has three more pins with a jumper between two of them. These pins are for selecting the trigger modes. The first one is called "non-repeatable trigger" and works like this: when the sensor output is high and the delay time is over, the output will high to low level. The other mode called "repeatable trigger" will automatically change fro keep the output high all the time until the detected object is present in sensor's range.

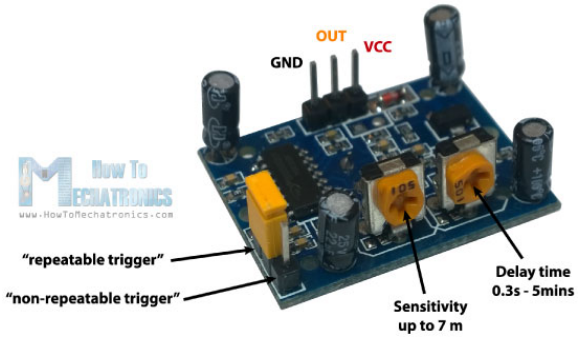

Figure 3. PIR Pin-Out

B. Relay Working Mechanism:

2)Overview

We can control High Voltage electronic devices using relays. A Relay is actually a switch which is electrically operated by an electromagnet. The electromagnet is activated with a low voltage, for example 5 volts from a microcontroller and it pulls a contact to make or break a high voltage circuit.

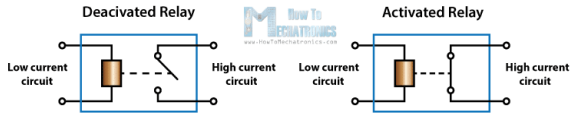

Figure 4. Conceptual Diagram

3)Circuit Schematic

For better understanding let's see the circuit schematics of the relay module in this configuration. So we can see that the 5 volts from our microcontroller connected to the Vcc pin for activating the relay through the Optocoupler IC are also connected to the JDVcc pin which powers the electromagnet of the relay.

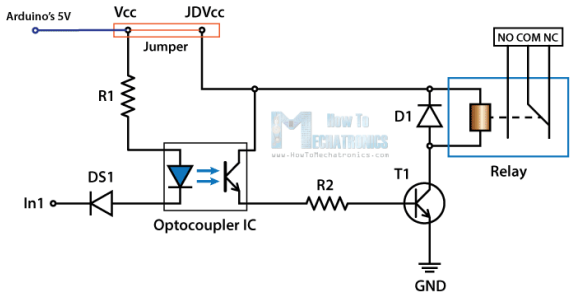

Figure 5. Relay Circuit Diagram

IV. BLUETOOTH MODE:

The Arduino Uno uses Universal Synchronous Asynchronous Receiver and Transmitter (USART) for communication between our computer and any other serial device. It uses a baud rate of 9600. So to access the wireless portion of our object we use the HC-05 Bluetooth Module which will hook up our Arduino's serial USART port to the Bluetooth communication in our phone or our laptop using an apps called BlueTerm and TeraTerm respectively.

V.RASPBERRY PI AS HUB

Raspberry pi is a small credit card sized computer. Which runs on debian linux. Here is where the internet of things concept shines. Our raspberry pi is connected to the arduino through its general purpose input/output pins where it allows us to check if there the PIR Sensor is activated or not , telling us if there is motion or not and sending this to data through the web to the Amazon Web Services Cloud

VI. CONNECTING TO AWS IOT

The Raspberry pi is connected to AWS IoT platform provided by Amazon. AWS IoT acts as hub where we can receive sensor data and send to his data to other AWS services for further processing. We can send it to AWS's NoSQL database, Dynamodb where it can store the data. We can send it to AWS's Machine Kenesis Engine to run programs based on the conditions of the sensor. Here is this project we us AWS's Simple Notification Service where we get an email whenever theres is motion notifying us if there is an intruder.

B. Connecting to AWS IoT

To connect the Raspberry Pi to AWS IoT sign in to Amazon Web Services website(aws.amazon.com) and go to AWS IoT console. Here we must create a thing under registry and follow the steps as it prompts. This will download certificates an AWS IoT

SDK for Python. AWS IoT uses MQTT protocol to communicate with devices.

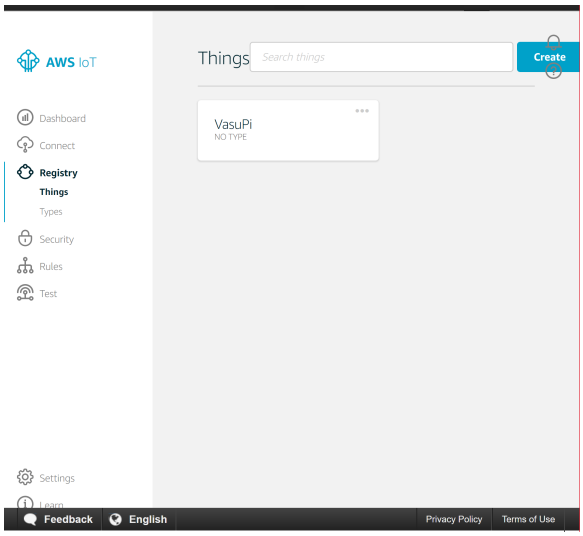

Figure 6. Creating a Thing

Now move over to Simple Notification Service Console and create a topic and follow the prompts told. Now the Raspberry Pi is connected to AWS IoT and we can now create a rule in AWS IoT console to send us notifications through Simple Notification Service whenever there is motion.

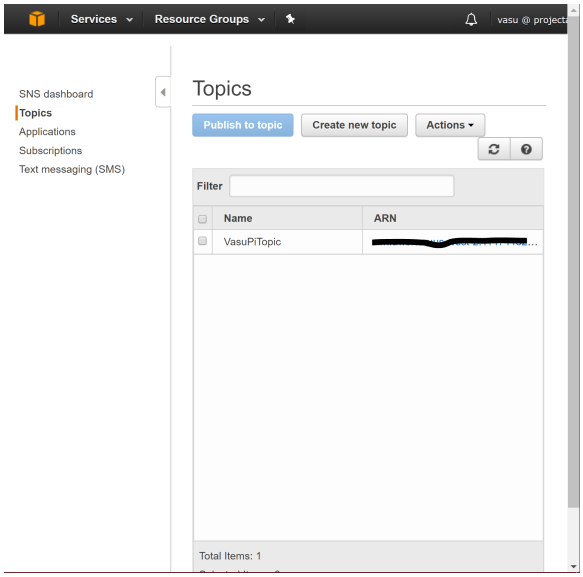

Figure 7. Creating a topic

VII. CODE FOR THE PROJECT:

A. Arduino Code

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

char data;
char modedata;
int pir=8;
int relay=9;
int value=A0;
int temp=0;
float volts=0.0;

void setup() {
pinMode(relay,OUTPUT);
pinMode(pir,INPUT);
lcd.begin(16,2);
Serial.begin(9600);
}

void loop() {
 lcd.setCursor(0,0);
 lcd.print("Bluetooth Mode");
 value=analogRead(A0); //read from A0
       volts=(value/1024.0)*5.0;
//conversion to volts
       temp=volts*100.0;
//conversion to temp Celsius


//display temp no lcd
  lcd.setCursor(0,1);
  lcd.print("TEMP= ");
  lcd.print(temp);
  lcd.print(" C");
modedata = Serial.read(); //Read byte
of data
//Turn Relay on
if (modedata == '1')
{
digitalWrite(relay,LOW);
Serial.println("Lamp ON");
}
//Turn Lamp off
else if (modedata == '0')
{
digitalWrite(relay, HIGH);
Serial.println("Lamp OFF");
}
else if (modedata == '5')
{lcd.clear();
  lcd.print("Motion Detection Mode");
   value=analogRead(A0);          //read
from A0
              volts=(value/1024.0)*5.0;
//conversion to volts
               temp=       volts*100.0;
//conversion to temp Celsius


//display temp no lcd
```

```
  lcd.setCursor(0,1);
  lcd.print("TEMP= ");
  lcd.print(temp);
  lcd.print(" C");
  while(modedata == '5')
  { int sensorValue = digitalRead(pir);

  if (sensorValue == 0)
  {
    digitalWrite(relay,HIGH);
    digitalWrite(3,LOW)
  }
else   if (sensorValue == 1) {

      digitalWrite(relay, LOW); // The
Relay Input works Inversely
    digitalWrite (3,HIGH);
    delay(500);
}
if(Serial.available()>0)
{
   Serial.println("exited");
   lcd.clear();
  break;
 }
}
}
}
```

B. Raspberry Pi Code for AWS IoT

```
from   AWSIoTPythonSDK.MQTTLib   import
AWSIoTMQTTClient
import sys
import logging
import time
import getopt
import RPi.GPIO as gpio
import json

# To read sensor data
gpio.setmode(gpio.BCM)
gpio.setup(18,gpio.IN)

def pir_state():
input_state =
if input_state == True:
state = "Intruder alert"
print("intruder alert")
time.sleep(1)
elif input_state == False:
state ="No one home"
print("No one home")
time.sleep(1)
return state
# 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")

# Usage
usageInfo = """Usage:

Use    certificate    based    mutual
authentication:
python basicPubSub.py -e <endpoint> -r
<rootCAFilePath> -c <certFilePath> -k
<privateKeyFilePath>

Use MQTT over WebSocket:
python basicPubSub.py -e <endpoint> -r
<rootCAFilePath> -w

Type  "python  basicPubSub.py  -h"  for
available options.
"""
# Help info
helpInfo = """-e, --endpoint
      Your AWS IoT custom endpoint
-r, --rootCA
      Root CA file path
-c, --cert
      Certificate file path
-k, --key
      Private key file path
-w, --websocket
      Use MQTT over WebSocket
-h, --help
      Help information


"""

# Read in command-line parameters
useWebsocket = False
host = ""
rootCAPath = ""
certificatePath = ""
privateKeyPath = ""
try:
      opts,           args           =
getopt.getopt(sys.argv[1:],
"hwe:k:c:r:",    ["help",    "endpoint=",
"key=","cert=","rootCA=", "websocket"])
      if len(opts) == 0:
            raise
getopt.GetoptError("No             input
parameters!")
      for opt, arg in opts:
            if    opt    in    ("-h",
"--help"):
      print(helpInfo)
```

```
        exit(0)
                if     opt     in     ("-e",
"--endpoint"):
        host = arg
                if     opt     in     ("-r",
"--rootCA"):
        rootCAPath = arg
                if     opt     in     ("-c",
"--cert"):
        certificatePath = arg
                if opt in ("-k", "--key"):
                        privateKeyPath     =
arg
                if     opt     in     ("-w",
"--websocket"):
        useWebsocket = True
except getopt.GetoptError:
        print(usageInfo)
        exit(1)

# Missing configuration notification
missingConfiguration = False
if not host:
        print("Missing       '-e'       or
'--endpoint'")
        missingConfiguration = True
if not rootCAPath:
        print("Missing       '-r'       or
'--rootCA'")
        missingConfiguration = True
if not useWebsocket:
        if not certificatePath:
                print("Missing    '-c'    or
'--cert'")
        missingConfiguration = True
        if not privateKeyPath:
                print("Missing    '-k'    or
'--key'")
        missingConfiguration = True
if missingConfiguration:
        exit(2)

# Configure logging
logger                                    =
logging.getLogger("AWSIoTPythonSDK.core
")
logger.setLevel(logging.DEBUG)
streamHandler = logging.StreamHandler()
formatter                                 =
logging.Formatter('%(asctime)s          -
%(name)s      -      %(levelname)s      -
%(message)s')
streamHandler.setFormatter(formatter)
logger.addHandler(streamHandler)

# Init AWSIoTMQTTClient
myAWSIoTMQTTClient = None
if useWebsocket:
        myAWSIoTMQTTClient=
AWSIoTMQTTClient("basicPubSub",useWebso
cket=True)

        myAWSIoTMQTTClient.configureEndpoint(ho
st, 443)

        myAWSIoTMQTTClient.configureCredentials
(rootCAPath)
else:
        myAWSIoTMQTTClient                =
AWSIoTMQTTClient("basicPubSub")

        myAWSIoTMQTTClient.configureEndpoint(ho
st, 8883)

        myAWSIoTMQTTClient.configureCredentials
(rootCAPath,              privateKeyPath,
certificatePath)

        #       AWSIoTMQTTClient       connection
configuration
        myAWSIoTMQTTClient.configureAutoReconne
ctBackoffTime(1, 32, 20)
        myAWSIoTMQTTClient.configureOfflinePubl
ishQueueing(-1)
        myAWSIoTMQTTClient.configureDrainingFre
quency(2) # Draining: 2 Hz
        myAWSIoTMQTTClient.configureConnectDisc
onnectTimeout(10) # 10 sec
        myAWSIoTMQTTClient.configureMQTTOperati
onTimeout(5) # 5 sec

        # Connect and subscribe to AWS IoT
        myAWSIoTMQTTClient.connect()
        myAWSIoTMQTTClient.subscribe("sdk/test/
intrudertest", 1, customCallback)
        time.sleep(2)

        # Publish to the same topic in a loop
forever
        loopCount = 0
        while True:
        s = pir_state()
        messageObject = {"motionstate" : s}
        message = json.dumps(messageObject)
        if s == "Intruder alert":
        myAWSIoTMQTTClient.publish("sdk/test/in
trudertest", message,1)
        loopCount += 1
        time.sleep(5)
        elif
                s == "No one home":
        print(s)
```

VIII. IMAGES

Figure 8. Prototype

Figure 9.. Prototype

IX. CONCLUSION

This project shows how a house staple like table lamp can be made modern, futuristic with the concept of internet of things and seeing how connected appliances can ease our life.

X.REFERENCES

[1] howtomechatronics.com
[2] arduino.cc
[3] Exploring Arduino by Jeremy Blum
[4] aws.amazon.com
[5] docs.aws.amazon.com
[6] w3schools.com