Course on Designing and Implementation of IoT

Designed and Conceptualized
By
Prof.Dr.Saurabh Mehta*
With
Ms. Michelle D'Souza#
Ms. Prashasti Sar

*Department of Electronics and Telecommunication
Vidyalankar Institute of Technology,Mumbai
Mahindra Susten Pvt. Ltd, Mumbai

saurabh.n.mehta@gmail.com

Acknowledgement

This Project was funded and supported by Global Power Source Pvt.Ltd., Mumbai

1	General Information
Proposal Title	Course On: Designing and Implementation of IoT
Index Terms	IOT, WSN, Cloud Storage, Interfacing , Sensors, Integration of Services
Disciplines	Electronics, Telecommunication, Computer Science, Process Control , Automation, etc
Introduction	The explosive growth of the "Internet of Things" is changing our world and the rapid drop in price for typical IoT components is allowing people to innovate new designs and products at home. In this course you will learn the importance of IoT, the current status of IoT devices and trends for the future. IoT design considerations, constraints and interfacing between the physical world and your device will also be covered. You will also learn how to make design trade-offs between hardware and software. We'll also cover key components of networking to ensure that participants understand how to connect their device to the Internet.
Objectives	• To understand the difference between current networking technologies and IoT • To learn the fundamentals of IoT • To design and Implement the IoT • To develop the customized solution.
Duration:	70 Hrs to 80 Hrs
Course Designer	 Dr.Saurabh N. Mehta has more than 15 years of professional, research, and teaching experience in wireless system development all together. Dr.Saurabh N.Mehta is currently working as a professor in the department of electronics and telecommunication at Vidyalankar Institute of Technology, Mumbai. Dr. Saurabh N. Mehta is responsible for integrating academia and industry collaboration, research program, planning, and educational training for the department of electronics and telecommunications. Dr. Saurabh N.Mehta earned his B.E., M.S., and PhD. degrees in Electronics Engineering from Mumbai University, Mumbai, India, Ajou University, South Korea, and Inha University, South Korea, in 2002, 2005, and 2011, respectively. He has over 65 research articles published in books, journals, national and international conferences, and several technical contributions to IEEE standards. He has been a reviewer, technical committee member, and editorial board for many international/national conferences and journals. He has won several prizes for the technical papers and projects. He is member of many international and national professional organizations including fellow of IETE, Senior Member of IEEE , Senior Life Members of CSI, Senior Member of IACSIT .Singapore, life members of ISTE to name a few. He has trained more than 2000 teachers/Students in the field of Electronics and Telecommunication through various guest lectures/seminars/workshops. His research interests are in the area of Wireless

	Networks, Rf-Id Technology, Ubiquitous Computing, Game Theory, and Educational Technology. He is also running his own "Sun Wireless Technologies", a consultancy firm in the area of embedded System, wireless technology, education, and technical market research. Beside technology he loves to spend his time in watching movies, reading books and educating people about HAM Radio.  Ms. Michelle D'souza has more than 1 year of professional experience in the area of embedded system designing. Currently she is working as Junior Product Engineer at Mahindra Susten Pvt. Ltd. (MachinePulse), Mumbai. Her job profile includes product development, system designing, and preparing feasibility reports. She earned her B.E. degree in Electronics and Telecommunication from Mumbai University in year 2016. Her areas of interest are in environment research, product development and educating students on technology. Throughout her academic career she has been part of many training and workshop programs and also won several prizes in technical events and sports. Beside technology she loves to play music and football.
Course Collaborator	Ms. Prashasti Sar
Teaching Mode	Self-Learning
Resources Needed	Hardware Boards and Sensors
Email	saurabh.n.mehta@gmail.com;13.michelledsouza@gmail.com
Tel:	0091-9833955002 (6 pm ~ 9 pm)
Fax:	N/A
Code Link	https://drive.google.com/drive/folders/1etLbB6rPdOd08cGogMFRQwK4N_avELDs?usp=sharing

Use this QR code to download IoT Course with Codes

Preface

I am very pleased to release self-learning hands- on course on "Designing and Implementation of IoT". This course consists of 49 experiments with different difficulty levels. Experiments are designed in such a way that students can learn all necessary skills to implement end to end IoT solution. All experiments are performed and successfully implemented on hardware to assure the quality of learning material. All experiments are separated by board types and categorized in three domains: sensor integration, implementation of communication protocols, and software integration. On completion of this course students will be able to understand the practical aspects of IoT designing and implementation. This course will help students to develop their programming, hardware integration, and embedded engineering skills, and hence employability in industry.

I would like to take this opportunity to thank Ms. Michelle D'Souza and Ms. Prashasti Sar in developing and implementing this course. Also, I would like to thank Global Power Source Pvt. Ltd., for their generous financial support in developing this course.

I hope this course will provide a good start to students and faculty members who would like to work in the field of IoT and cultivate entrepreneurship spirit in them.

I would be happy to know your suggestions and feedbacks to improve the presentation and contents of this course.

Happy Learning!

Prof.Dr.Saurabh N. Mehta
Professor,
Department of Electronics and Telecommunication,
Vidyalankar Institute of Technology, Mumbai.

Course Content: Designing and Implementation of IoT

Sr.No.	Module	Topic	Content	Difficulty Level
1.	Module –I	Arudino Board	Experiment 1 to 30	Easy & Moderate
2.	Module-II	ATtiny85 Board	Experiment 31 to 33	Moderate
3.	Module-III	ESP8266 Board	Experiment 34 to 37	Moderate
4.	Module-IV	NodeMCU Board	Experiment 38 to 40	Easy & Moderate
5.	Module-V	Raspberry Pi Board	Experiment 41 to 46	Moderate
6.	Module-VI	Arudino-Raspberry Pi Boards	Experiment 47 & 48	Moderate
7.	Module-VII	Software	Experiment 49	Moderate

Index

Hardware List

Sr.No	Components	Tentaive Price
1	Arduino Yun	7000
2	RC522 Rfid Reader/Writer with Cards	250
3	DHT11 Temp and Humidity Sensor Module	200
4	Ultrasonic Module	200
5	Arduino	1500
6	Xbee Breakout Board	100
7	Rs485 Module	150
8	Sharp analog Distance Sensor	500
9	Sharp Digital Distance Sensor	600
10	5v GPS Receiver with Antenna	1775
11	IR receiver Diode	120
12	FTDI Basic Breakout	500
13	Tracking Shield	3499
14	Waterflow sensor	350
15	Rs485 Module	200
16	5v Buzzer	150
17	OLED Display	350
18	Pulse Sensor	400
19	DS1307 RTC Module	120
20	VM226 USB to RS485	200
21	VM381 Lithium Charging Module	120
22	12v lead acid	800
23	2 pin connector	50
24	MQ135	450
25	Arduino Pro Mini	250
26	NodeMCU	400
27	L293d Motor Driver	150
28	Dual Speed Encoder	350
29	Laser	10
30	Stepper Motor BO	200
31	BO DC wheel yellow	50
32	MQ2	180
33	Solar Panel	350
34	VM343 Digispark Attiny	170
35	MB-102 Breadboard	250
36	Bluetooth module	300
37	Breadboard PS	100
38	IR obstacle sensor	120
39	RFID cards	600
40	Arduino Uno	500
41	Esp8266	250
42	RPI 3	2300
43	RPI 3 camera	100
44	Xbee	1200
45	Xbee explorer USB	340
46	USB to RS232 cable	150

Experiment 1: Calibrating Polulu analog distance sensor

Aim:

To calibrate the Polulu analog distance sensor.

Objective:

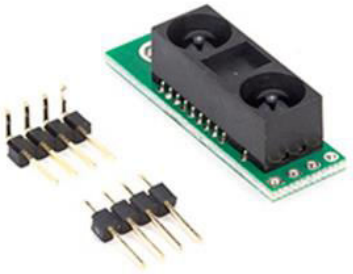
To learn how to calibrate sensor to produce accurate readings.

Components Required:

1. Arduino UNO
2. Pololu Carrier With Sharp GP2Y0A60SZLF Analog Distance Sensor
3. Connecting wires

Connections:

Arduino Pins	GP2Y0A60SZLF
5v	Vcc
Gnd	Gnd
Out	Out
Leave open	Enable



Procedure:

1. In order to log data to excel sheet from Arduino, we need to first download and install external software.
2. Please go to reference 1 and download and install the software on your computer according to the specifications of your system. (Windows, Linux, Mac, etc.)
3. Now open the Application just installed *PLX-DAQ* and select the COM port number and set Baud rate to 9600.
4. Next connect the analog sensor and upload the source code given below.
5. Place the obstacle 10cm away from the sensor.
6. After every 10 seconds, make sure to move the obstacle 2cm away from the sensor till you reach 100cm.

7. Save your excel sheet and select the Sensor value and Distance columns to create a graph.
8. Chose power trend and display the formula in excel.
9. That's all. Use this formula to get the correct distance value.

Source Code:

```
/**
  Code modified by : Ms. Michelle D'Souza, IoT Intern, 2016.
 //http://www.instructables.com/id/Sending-data-from-Arduino-to-Excel-and-plotting-it/?ALLSTEPS
 //http://www.instructables.com/id/How-to-setup-a-Pololu-Carrier-with-Sharp-GP2Y0A60S/?ALLSTEPS
 //always starts in line 0 and writes the thing written next to LABEL
**/

long sensorsum = 0;
int n = 1;
int mean = 0;
float distance = 6;

void setup() {

 Serial.begin(9600); // the bigger number the better

 Serial.println("CLEARDATA"); //clears up any data left from previous projects

 Serial.println("LABEL,Time,Timer,Distance,Sensor Value,Mean"); //always write LABEL, so excel knows the
next things will be the names of the columns (instead of Acolumn you could write Time for instance)

 Serial.println("RESETTIMER"); //resets timer to 0

}

void loop() {

 int sensorValue = analogRead(A0);
 sensorsum = (sensorsum + sensorValue);
 mean = (sensorsum / n);
 n = n + 1;

 delay(50); //add a delay

 Serial.print("DATA,TIME,TIMER,"); //writes the time in the first column A and the time since the
measurements started in column B
 Serial.print(distance);
 Serial.print(",");
 Serial.print(sensorValue);
 Serial.print(",");
 Serial.print(mean);
 Serial.println(); //be sure to add println to the last command so it knows to go into the next row on the
second run

 if (n > 25) {
  digitalWrite(13, HIGH);
  delay(4000);
```

```
  digitalWrite(13, LOW);
  distance = distance + 2 ;
  n = 1;
  sensorsum = 0;
  mean = 0;
 }

}
```

Applications:

- Data acquisition system: The data obtained from any sensor is important in case of IoT applications. This data obtained can be continuously stored in excel sheet of computer for future reference.

Conclusion:

The sensor is calibrated to give accurate values of distance.

References:

[1] https://www.parallax.com/downloads/plx-daq
[2] http://www.micropik.com/PDF/HCSR04.pdf
[3] http://www.instructables.com/id/Sending-data-from-Arduino-to-Excel-and-plotting-it/?ALLSTEPS

Experiment 2: Working with MQ2 and MQ135 Gas sensor

Aim:

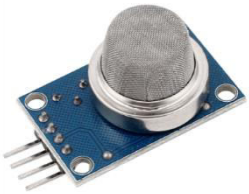
To extract information from gas sensors using Arduino.

Objective:

To learn how to use MQ Gas sensors and extract data from it.

Components Required:

1. Arduino and power supply
2. MQ135 Gas Sensor
3. MQ2 Gas Sensor
4. Connecting wires.



Connections:

Connections of MQ2:

Arduino Pins	MQ2
5v	VCC
Gnd	Gnd
A0	A0

Connections of the MQ135:

Arduino Pins	MQ135
5v	VCC
Gnd	Gnd
A1	A0

Do the connections as shown in the table.

Theory:

- Is is better to use the modules instead of the component itself as we can change the sensitivity.

- MQ-2

  Sensitive for Methane, Butane, LPG, smoke.
  This sensor is sensitive for flammable and combustible gasses.
  The heater uses 5V.

- MQ135

  For Air Quality.
  Sensitive for Benzene, Alcohol, smoke.
  The heater uses 5V.

- The output voltage from the Gas sensor increases when the concentration of gas increases. Sensitivity can be adjusted by varying the potentiometer.

Procedure:

Once you've set up your connections with the Arduino, complete these steps before uploading the code on the Arduino.

1. Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows to read information from both the sensors using Arduino.
2. Go to link provided in Reference [1] and Reference [2] and download the libraries.
3. Next in the Arduino IDE, click on :
4. *Sketch >> Include Library >> Add .ZIP library.*
5. Browse and select the folder where you downloaded the libraries.
6. Once you've added both the libraries to the Arduino, you must power the Gas Sensors for at least 6 hours before extracting information. The pre-heat period is 24 hours in the datasheet but 6 hours will work fine.
7. We are good and ready to go. Create a new sketch and copy paste the code contents provided below.
8. Make sure you shot the ground of external power supply and Arduino if you're using it.
9. Upload the code and view the ppm of gases on serial monitor.

Source Code:

```
/* MQ2 and MQ135
```

```
  Code modified by: Ms.Michelle D'Souza, IoT Intern, 2016
*/

/*******************Demo for MQ-2 Gas Sensor Module
V1.0*****************************
  Support:  Tiequan Shao: support[at]sandboxelectronics.com

  Lisence: Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA
3.0)

  Note:    This piece of source code is supposed to be used as a
demostration ONLY. More
         sophisticated calibration is required for industrial field
application.
  http://sandboxelectronics.com/?p=165
                                                      Sandbox
Electronics    2011-04-25
************************************************************************
*********/

/************************Hardware Related
Macros************************************/
#define         MQ_PIN                          (0)     //define which analog
input channel you are going to use
#define         RL_VALUE                        (5)     //define the load
resistance on the board, in kilo ohms
#define         RO_CLEAN_AIR_FACTOR             (9.83)  //RO_CLEAR_AIR_FACTOR=
(Sensor resistance in clean air)/RO,
//which is derived from the chart in datasheet

/************************Software Related
Macros************************************/
#define         CALIBARAION_SAMPLE_TIMES        (50)    //define how many
samples you are going to take in the calibration phase
#define         CALIBRATION_SAMPLE_INTERVAL     (500)   //define the time
interal(in milisecond) between each samples in the
//cablibration phase
#define         READ_SAMPLE_INTERVAL            (50)    //define how many
samples you are going to take in normal operation
#define         READ_SAMPLE_TIMES               (5)     //define the time
interal(in milisecond) between each samples in
//normal operation
#include "MQ135.h"
#define ANALOGPIN A1    //  Define Analog PIN on Arduino Board
#define RZERO 206.85    //  Define RZERO Calibration Value
MQ135 gasSensor = MQ135(ANALOGPIN);
/*********************Application Related
Macros**********************************/
#define         GAS_LPG                         (0)
#define         GAS_CO                          (1)
#define         GAS_SMOKE                       (2)

/****************************Globals**********************************
*********/
float           LPGCurve[3]  =  {2.3, 0.21, -0.47}; //two points are taken
from the curve.
//with these two points, a line is formed which is "approximately
equivalent"
//to the original curve.
```

```
//data format:{ x, y, slope}; point1: (lg200, 0.21), point2: (lg10000, -
0.59)
float           COCurve[3]  =  {2.3, 0.72, -0.34};  //two points are taken
from the curve.
//with these two points, a line is formed which is "approximately
equivalent"
//to the original curve.
//data format:{ x, y, slope}; point1: (lg200, 0.72), point2:
(lg10000,  0.15)
float           SmokeCurve[3] = {2.3, 0.53, -0.44}; //two points are taken
from the curve.
//with these two points, a line is formed which is "approximately
equivalent"
//to the original curve.
//data format:{ x, y, slope}; point1: (lg200, 0.53), point2: (lg10000,  -
0.22)
float           Ro           =  10;                 //Ro is initialized to
10 kilo ohms

void setup()
{
  Serial.begin(9600);                               //UART setup, baudrate
= 9600bps
  Serial.print("Calibrating...\n");
  Ro = MQCalibration(MQ_PIN);                       //Calibrating the
sensor. Please make sure the sensor is in clean air
  //when you perform the calibration
  Serial.print("Calibration is done...\n");
  Serial.print("Ro=");
  Serial.print(Ro);
  Serial.print("kohm");
  Serial.print("\n");

  float rzero = gasSensor.getRZero();
  delay(3000);
  Serial.print("MQ135 RZERO Calibration Value : ");
  Serial.println(rzero);
}

void loop()
{
  Serial.print("LPG:");
  Serial.print(MQGetGasPercentage(MQRead(MQ_PIN) / Ro, GAS_LPG) );
  Serial.print( "ppm" );
  Serial.print("    ");
  Serial.print("CO:");
  Serial.print(MQGetGasPercentage(MQRead(MQ_PIN) / Ro, GAS_CO) );
  Serial.print( "ppm" );
  Serial.print("    ");
  Serial.print("SMOKE:");
  Serial.print(MQGetGasPercentage(MQRead(MQ_PIN) / Ro, GAS_SMOKE) );
  Serial.print( "ppm" );
  Serial.print("\n");
  delay(200);

  float ppm = gasSensor.getPPM();
  delay(1000);
  digitalWrite(13, HIGH);
  Serial.print("CO2 ppm value : ");
```

```
  Serial.println(ppm);
}

/***************** MQResistanceCalculation
****************************************
  Input:   raw_adc - raw value read from adc, which represents the voltage
  Output:  the calculated sensor resistance
  Remarks: The sensor and the load resistor forms a voltage divider. Given
the voltage
         across the load resistor and its resistance, the resistance of the
sensor
         could be derived.
************************************************************************
*********/
float MQResistanceCalculation(int raw_adc)
{
  return ( ((float)RL_VALUE * (1023 - raw_adc) / raw_adc));
}

/***************************** MQCalibration
****************************************
  Input:   mq_pin - analog channel
  Output:  Ro of the sensor
  Remarks: This function assumes that the sensor is in clean air. It use
         MQResistanceCalculation to calculates the sensor resistance in
clean air
         and then divides it with RO_CLEAN_AIR_FACTOR. RO_CLEAN_AIR_FACTOR
is about
         10, which differs slightly between different sensors.
************************************************************************
*********/
float MQCalibration(int mq_pin)
{
  int i;
  float val = 0;

  for (i = 0; i < CALIBARAION_SAMPLE_TIMES; i++) {      //take multiple
samples
    val += MQResistanceCalculation(analogRead(mq_pin));
    delay(CALIBRATION_SAMPLE_INTERVAL);
  }
  val = val / CALIBARAION_SAMPLE_TIMES;                 //calculate the
average value

  val = val / RO_CLEAN_AIR_FACTOR;                      //divided by
RO_CLEAN_AIR_FACTOR yields the Ro
  //according to the chart in the datasheet

  return val;
}
/*****************************  MQRead
*****************************************
  Input:   mq_pin - analog channel
  Output:  Rs of the sensor
  Remarks: This function use MQResistanceCalculation to caculate the sensor
resistenc (Rs).
         The Rs changes as the sensor is in the different consentration of
the target
```

```
         gas. The sample times and the time interval between samples could
be configured
         by changing the definition of the macros.
************************************************************************
*********/
float MQRead(int mq_pin)
{
  int i;
  float rs = 0;

  for (i = 0; i < READ_SAMPLE_TIMES; i++) {
    rs += MQResistanceCalculation(analogRead(mq_pin));
    delay(READ_SAMPLE_INTERVAL);
  }

  rs = rs / READ_SAMPLE_TIMES;

  return rs;
}

/*****************************  MQGetGasPercentage
**********************************
  Input:   rs_ro_ratio - Rs divided by Ro
           gas_id      - target gas type
  Output:  ppm of the target gas
  Remarks: This function passes different curves to the MQGetPercentage
function which
         calculates the ppm (parts per million) of the target gas.
************************************************************************
*********/
int MQGetGasPercentage(float rs_ro_ratio, int gas_id)
{
  if ( gas_id == GAS_LPG ) {
    return MQGetPercentage(rs_ro_ratio, LPGCurve);
  } else if ( gas_id == GAS_CO ) {
    return MQGetPercentage(rs_ro_ratio, COCurve);
  } else if ( gas_id == GAS_SMOKE ) {
    return MQGetPercentage(rs_ro_ratio, SmokeCurve);
  }

  return 0;
}

/*****************************  MQGetPercentage
**********************************
  Input:   rs_ro_ratio - Rs divided by Ro
           pcurve      - pointer to the curve of the target gas
  Output:  ppm of the target gas
  Remarks: By using the slope and a point of the line. The x(logarithmic
value of ppm)
         of the line could be derived if y(rs_ro_ratio) is provided. As it
is a
         logarithmic coordinate, power of 10 is used to convert the result
to non-logarithmic
         value.
************************************************************************
*********/
int  MQGetPercentage(float rs_ro_ratio, float *pcurve)
{
```

```
  return (pow(10, ( ((log(rs_ro_ratio) - pcurve[1]) / pcurve[2]) +
pcurve[0])));
}
```

Applications:

- Real Time Air Monitoring system.

Conclusion:

With the use of Arduino and MQ sensors and based on previous projects the data can be sent over the internet to thngspeak channel and an Iot based Air Monitoring System can be created.

References:

[1] https://github.com/ckalpha/MQ135

[2] http://sandboxelectronics.com/?p=165

[3] http://davidegironi.blogspot.in/2014/01/cheap-co2-meter-using-mq135-sensor-with.html#.WJHc4VV97IW

[4] http://playground.arduino.cc/Main/MQGasSensors

[5] https://www.eprolabs.com/wp-content/uploads/2016/07/SEN-0049.pdf

[6] http://wiki.seeed.cc/Grove-Gas_Sensor-MQ2/

Experiment 3: Measuring flow of water with Arduino

Aim:

To extract information from water flow sensor using Arduino.

Components Required:

1. Arduino and power supply
2. Water Flow Sensor
3. Connecting wires.

Connections:

Connections of MQ2:

Arduino Pins	Water Flow Sensor
5v	VCC
Gnd	Gnd
A0	A0

Do the connections as shown in the table.

Theory:

A very highly detailed description of the sensor is done in the link given in Reference [1].

Procedure:

Once you've set up your connections with the Arduino, complete these steps before uploading the code on the Arduino.

10. Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows to read information from both the sensors using Arduino.
11. We are good and ready to go. Create a new sketch and copy paste the code contents provided below.
12. Make sure you shot the ground of external power supply and Arduino if you're using it.
13. Upload the code and view the ppm of gases on serial monitor.

Source Code:

```
/*
  Liquid flow rate sensor -DIYhacking.com Arvind Sanjeev
  https://diyhacking.com/arduino-flow-rate-sensor/
  Measure the liquid/water flow rate using this code.
  Connect Vcc and Gnd of sensor to arduino, and the
  signal line to arduino digital pin 2.
*/

byte statusLed    = 13;
byte sensorInterrupt = 0;  // 0 = digital pin 2
byte sensorPin       = 2;

// The hall-effect flow sensor outputs approximately 4.5 pulses per second
per
// litre/minute of flow.
float calibrationFactor = 4.5;

volatile byte pulseCount;

float flowRate;
unsigned int flowMilliLitres;
unsigned long totalMilliLitres;

unsigned long oldTime;

void setup()
{

  // Initialize a serial connection for reporting values to the host
  Serial.begin(38400);

  // Set up the status LED line as an output
  pinMode(statusLed, OUTPUT);
  digitalWrite(statusLed, HIGH);  // We have an active-low LED attached

  pinMode(sensorPin, INPUT);
  digitalWrite(sensorPin, HIGH);

  pulseCount        = 0;
  flowRate          = 0.0;
  flowMilliLitres   = 0;
  totalMilliLitres  = 0;
  oldTime           = 0;

  // The Hall-effect sensor is connected to pin 2 which uses interrupt 0.
  // Configured to trigger on a FALLING state change (transition from HIGH
  // state to LOW state)
  attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
}

/**
   Main program loop
*/
void loop()
{
```

```
  if ((millis() - oldTime) > 1000)    // Only process counters once per
second
  {
    // Disable the interrupt while calculating flow rate and sending the
value to
    // the host
    detachInterrupt(sensorInterrupt);

    // Because this loop may not complete in exactly 1 second intervals we
calculate
    // the number of milliseconds that have passed since the last execution
and use
    // that to scale the output. We also apply the calibrationFactor to
scale the output
    // based on the number of pulses per second per units of measure
(litres/minute in
    // this case) coming from the sensor.
    flowRate = ((1000.0 / (millis() - oldTime)) * pulseCount) /
calibrationFactor;

    // Note the time this processing pass was executed. Note that because
we've
    // disabled interrupts the millis() function won't actually be
incrementing right
    // at this point, but it will still return the value it was set to just
before
    // interrupts went away.
    oldTime = millis();

    // Divide the flow rate in litres/minute by 60 to determine how many
litres have
    // passed through the sensor in this 1 second interval, then multiply
by 1000 to
    // convert to millilitres.
    flowMilliLitres = (flowRate / 60) * 1000;

    // Add the millilitres passed in this second to the cumulative total
    totalMilliLitres += flowMilliLitres;

    unsigned int frac;

    // Print the flow rate for this second in litres / minute
    Serial.print("Flow rate: ");
    Serial.print(int(flowRate));  // Print the integer part of the variable
    Serial.print(".");             // Print the decimal point
    // Determine the fractional part. The 10 multiplier gives us 1 decimal
place.
    frac = (flowRate - int(flowRate)) * 10;
    Serial.print(frac, DEC) ;      // Print the fractional part of the
variable
    Serial.print("L/min");
    // Print the number of litres flowed in this second
    Serial.print("  Current Liquid Flowing: ");             // Output
separator
    Serial.print(flowMilliLitres);
    Serial.print("mL/Sec");

    // Print the cumulative total of litres flowed since starting
```

```
    Serial.print("  Output Liquid Quantity: ");             // Output
separator
    Serial.print(totalMilliLitres);
    Serial.println("mL");

    // Reset the pulse counter so we can start incrementing again
    pulseCount = 0;

    // Enable the interrupt again now that we've finished sending output
    attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
  }
}

/*
  Insterrupt Service Routine
*/
void pulseCounter()
{
  // Increment the pulse counter
  pulseCount++;
}
```

Conclusion:

The flow rate of water is calculated using Water Flow Senor.

References:

[1] https://diyhacking.com/arduino-flow-rate-sensor/

Experiment 4: Control and calculate speed of motor

Aim:

To calculate speed of motor and control the speed using a potentiometer.

Objective:

To understand how to calculate speed of motor and change the speed accordingly.

Components Required:

1. Arduino UNO
2. Potentiometer
3. Motor Driver module L293d
4. IR Speed sensor module with encoder wheel
5. Jumper Wires
6. Gear motor

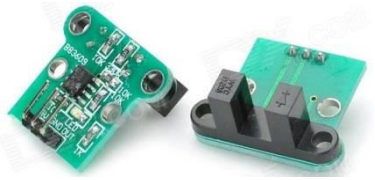
Connections:

Connections with the Potentiometer:

Arduino Pins	POT
3.3v	1
GND	3
8	2

Connections with the IR speed sensor module:

Arduino Pins	IR speed sensor
5v	5v
GND	GND
8	3



Connections with the Motor Driver module:

Arduino Pins	Motor Driver L293d
8	M1-IN
6	M1-IN

1. Perform the following connections as shown in the table.

2. Connect the gear motor to the M1-OUT and apply 9V DC voltage to the motor driver module.

Note: Make sure the GND of motor driver module is also connected to the GND of Arduino.

Procedure:

- Once you have your connections ready, copy paste the code given below and upload it to the Arduino.
- The speed of motor is measured by counting the number of times the IR beam of IR speed sensor is broken in a given interval of time.

  Formula:

$$Rps = \frac{Count}{Time(in\ seconds) \times No.of\ holes\ in\ Encoder\ Wheel}$$

  where

  Count : No. of times the IR beam breaks.

- View the speed of motor on serial monitor in Rotations per second (rps).
- Change the motor speed by rotating the potentiometer knob to increase or decrease the speed.
- *Debugging : Measure the voltage with the help of a multimeter across the Motor Driver module while rotating the Potentiometer. The multimeter reading should increase gradually.*

Source Code:

```
/*
  Code developed by : Ms. Michelle D'Souza

  Grateful to the whole open source community.
*/
int state = LOW;
int lastState = LOW;
float count = 0;
int start;
unsigned long previousMillis = 0;        // will store last time LED was
updated
```

```
// constants won't change :
const long interval = 10000;
int in1 = 6;
int in2 = 8;
int encoder = 3;

void setup() {
  Serial.begin(9600);
  Serial.println("___Control and measure speed of motor___");
  Serial.println("Rps value displayed in every 10 sec.");
  pinMode(encoder, INPUT);
  state = digitalRead(encoder);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
}

void loop() {
  int potvalue = analogRead(1);  // Potentiometer connected to Pin A1
  //Serial.println("motorspeed");
  int motorspeed = map(potvalue, 0, 680, 255, 0);
  // Serial.println(motorspeed);

  analogWrite(in1, motorspeed);  // set speed of motor (0-255)
  digitalWrite(in2, 1);  // set rotation of motor to Clockwise
  //Serial.println();


  if (state == HIGH && lastState == LOW) {
   count++;
    Serial.println(count);
  }
  lastState = state;
  state = digitalRead(encoder);

  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {

    // save the last time you resetted.
    previousMillis = currentMillis;
    Serial.print("Rotation per second : ");
    float rotation = (count / (20 * 10 ));
    Serial.println(rotation);
    Serial.println("Reseting");
    count = 0;
  }
}
```

Applications:

- Automated speed control of motor in industry.

Conclusion:

With the use of IR speed sensor module, the speed of motor can be calculated and controlled according to the application.

References:

[1] https://brainy-bits.com/tutorials/speed-sensor-with-arduino/

[2] http://www.circuitmagic.com/arduino/diy-digital-rpm-tachometer-with-arduino/

Experiment 5 : Simple Access Control for specific User using RFID chip and Arduino

Aim:

To create a simple access control system which allows entry to specific user and restricts entry to non-authorized user.

Objective:

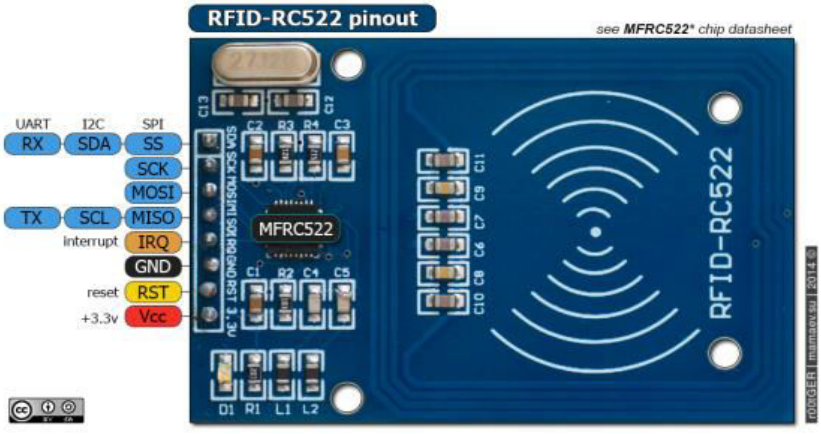
To learn about RFID and use RFID chip with Arduino board to create a simple access control system.

Components Required:

1. Arduino UNO
2. LED
3. Resistor 221ohm
4. RFID Chip: RC522 model
5. Jumper Wires

Connections:

Arduino Pins	RFID-RC522
Reset	9
SPI SS	10
SPI MOSI	11
SPI MISO	12
SPI SCK	13



1.Perform the following connections as shown in the table.
2. Connect the LED negative to GND of Arduino and positive to pin 9 with a resistance valued between 220Ω – 1KΩ.

Procedure:

Once you have your Adruino IDE installed and updated to the latest version, we will use an additional library which allows to read and write onto RFID cards using Arduino.

Go to link provided in Reference [2] and download the library.

Next in the Arduino IDE, click on :

Sketch >> Include Library >> Add .ZIP library.

Browse and select the folder where you downloaded the library.

Now, we have to read the RFID cards ID number in order to allow access to a specific card user only. So, in your arduino IDE click on:

File >> Examples >> MRFC522 >> ReadUidMultiReader.

Plug in your Arduino with the connections made to RFID Chip and upload this code. Open the Serial Monitor. Next, place your RFID cards close to the RFID reader and note down the UID numbers for both.

We are good and ready to go. Create a new sketch and copy paste the code contents provided below.

Make sure you change the UID number according to your card. Finally, hover your authorized card over the card reader and see the led glow!

Source Code:

```
/*This Code allows access to a specific user only using RFID chip RC522.
Code developed by Interns,2016.
*/
 /*
 * Pin layout should be as follows:
 * Signal     Pin
 *          Arduino Uno
 * -----------------------
 * Reset      9
 * SPI SS     10
 * SPI MOSI   11
 * SPI MISO   12
 * SPI SCK    13
 * Thanks to:https://github.com/miguelbalboa/rfid
 */

#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10
#define RST_PIN 9

int led = 9;
```

```
MFRC522 rfid(SS_PIN, RST_PIN); // Instance of the class

byte nuidPICC[3];// Init array that will store new NUID

void setup() {
 Serial.begin(9600);
 SPI.begin(); // Init SPI bus
 rfid.PCD_Init(); // Init MFRC522
 pinMode(13, OUTPUT);
 Serial.println(F("This code allows only authorised cards"));
}
/**
  Helper routine to dump a byte array as hex values to Serial.
*/
void printHex(byte *buffer, byte bufferSize) {
 for (byte i = 0; i < bufferSize; i++) {
  Serial.print(buffer[i] < 0x10 ? " 0" : " ");
  Serial.print(buffer[i], HEX);
 }
}

void loop() {
 //Change these numbers according to the Card ID you wanna allow.
 nuidPICC[0] = 0x35;
 nuidPICC[1] = 0x9A;
 nuidPICC[2] = 0xE2;
 nuidPICC[3] = 0x52;

 // Look for new cards
 if ( ! rfid.PICC_IsNewCardPresent())
  return;

 // Verify if the NUID has been read
 if ( ! rfid.PICC_ReadCardSerial())
  return;

 if (rfid.uid.uidByte[0] == nuidPICC[0] ||
    rfid.uid.uidByte[1] == nuidPICC[1] ||
    rfid.uid.uidByte[2] == nuidPICC[2] ||
    rfid.uid.uidByte[3] == nuidPICC[3] ) {

     Serial.println(F("Hi !Authorised card user!."));
     printHex(rfid.uid.uidByte, rfid.uid.size);
     //Instead of switching led on/off, a simple door open/close application can be implemented using relay.
     digitalWrite(led, HIGH);   // turn the LED on by making the voltage HIGH
     delay(1000);               // wait for a second
     digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
     delay(1000);               // wait for a second
    }

 else {
  Serial.println(F("Sorry.Not Authorised"));
  printHex(rfid.uid.uidByte, rfid.uid.size);
 }
}
```

Applications:

- RFID based entry system for employees in any organization.

Conclusion:

With the use of RFID chip and Arduino, easy access control for any application can be developed.

References:

[1] http://playground.arduino.cc/Learning/MFRC522

[2] https://github.com/miguelbalboa/rfid

Experiment 6 : IR Remote Control - RGB led

Aim:

To change the led color using IR remote blaster.

Objective:

To learn how to use RGB led and use IR blaster to control the color of RGB led.

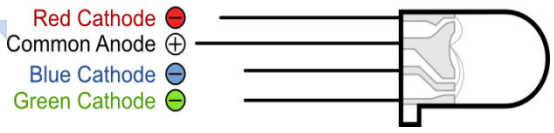
Components Required:

4. Arduino and power supply
5. RGB LED
6. Resistors : 330 ohm
7. Connecting wires.
8. IR Receiver Diode module TSOP38238

Connections:

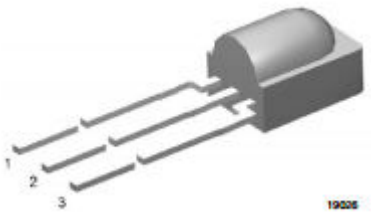
Connections of the RGB Led:

Arduino Pins	RGB Led
11	Red
5v	CA
10	Blue
9	Green



Connections of the IR Receiver Diode module:

Arduino Pins	IR Receiver Diode module
12	1
GND	2
3.3v	3



Pinning:
1 = OUT, 2 = GND, 3 = V_S

Perform the connections as shown in the table.

Procedure:

Once you've set up your connections with the Arduino, complete these steps before uploading the code on the Arduino.

14. Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows to read the IR pulse information from the IR blaster remote using Arduino.
15. Go to link provided in Reference [2] and download the library.
16. Next in the Arduino IDE, click on :
17. *Sketch >> Include Library >> Add .ZIP library.*
18. Browse and select the folder where you downloaded the library.
19. Now, we have to read the IR pulse information in order to change the color of LED. So, in your Arduino IDE click on:
20. *File >> Examples >> IRremote >> IRrecvDemo.*
21. Plug in your Arduino with the connections made to IR diode module and upload this code. Open the Serial Monitor. Next, point your IR remote towards the IR diode and note down the values occurring for various buttons pressed.
22. We are good and ready to go. Create a new sketch and copy paste the code contents provided below.
23. Make sure you change the values according to your Ir Remote. Finally, connect the RGB led and you can now control the color of led with the click of few buttons.
24. You can change the color of the LED, by changing the voltage to each pin. You can do this by changing the numbers in the Arduino code.

Source Code:

```
/*
   IRremote: IRrecvDemo - demonstrates receiving IR codes with IRrecv
   An IR detector/demodulator must be connected to the input RECV_PIN.
   Version 0.1 July, 2009
   Copyright 2009 Ken Shirriff
   https://cdn-learn.adafruit.com/downloads/pdf/adafruit-arduino-lesson-3-rgb-leds.pdf
   Code : IoT Intern, 2016
*/

#include <IRremote.h>

int RECV_PIN = 12;
int redPin = 11;
int bluePin = 10;
int greenPin = 9;
int result;
```

```
IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value,DEC);
    result=results.value,DEC;
    switch (result)
    {
      //Change the values according to your IR blaster remote.
      case 486:
        setColor(0, 255, 255); // red
        break;
      case 3558:
        setColor(255, 0, 255); // green
        break;
      case 2534:
        setColor(0, 0, 255); // yellow
        break;
      case 1510:
        setColor(80, 0, 80); // purple
        break;
            case 2471194969:
        setColor(0, 255, 255); // red
        break;
      case 3565633380:
        setColor(255, 0, 255); // green
        break;
      case 2427284352:
        setColor(0, 0, 255); // yellow
        break;
      case 592436055:
        setColor(80, 0, 80); // purple
        break;
    }
    irrecv.resume(); // Receive the next value
  }
  delay(100);
}

void setColor(int red, int green, int blue)
{
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

Applications:

- Easy wireless control of any appliance in industry or organization with the help of a relay.

Conclusion:

With the use of Arduino and any IR remote blaster used for TV, DVD, AC, etc., appliances can be wirelessly.

References:

[1] http://arcfn.com

[2] https://github.com/z3t0/Arduino-IRremote

[3] https://cdn-learn.adafruit.com/downloads/pdf/adafruit-arduino-lesson-3-rgb-leds.pdf

[4] https://www.sparkfun.com/datasheets/Sensors/Infrared/tsop382.pdf

Experiment 7 : To send ultrasonic sensor data to android device using bluetooth

Aim:

To get distance measurement using ultrasonic sensor and send it to an Android device using Bluetooth.

Objective:

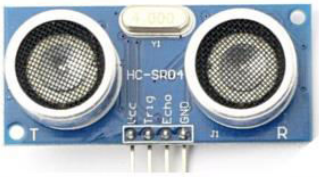
To be able to create useful of Internet of things based projects.

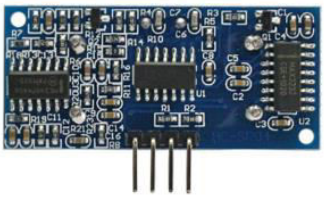
Components Required:

4. Arduino UNO
5. HC – SR04 ultrasonic sensor
6. HC 05 Bluetooth module

About the sensor:

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400 cm or 1" to 13 feet. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module.





Features

Power Supply :+5V DC

Quiescent Current : <2mA

Working Current: 15mA

Effectual Angle: <15°

Ranging Distance : 2cm – 400 cm/1" – 13ft

Resolution : 0.3 cm

Measuring Angle: 30 degree

Trigger Input Pulse width: 10uS

Dimension: 45mm x 20mm x 15mm

Pins

VCC: +5VDC

Trig : Trigger (INPUT)

Echo: Echo (OUTPUT)

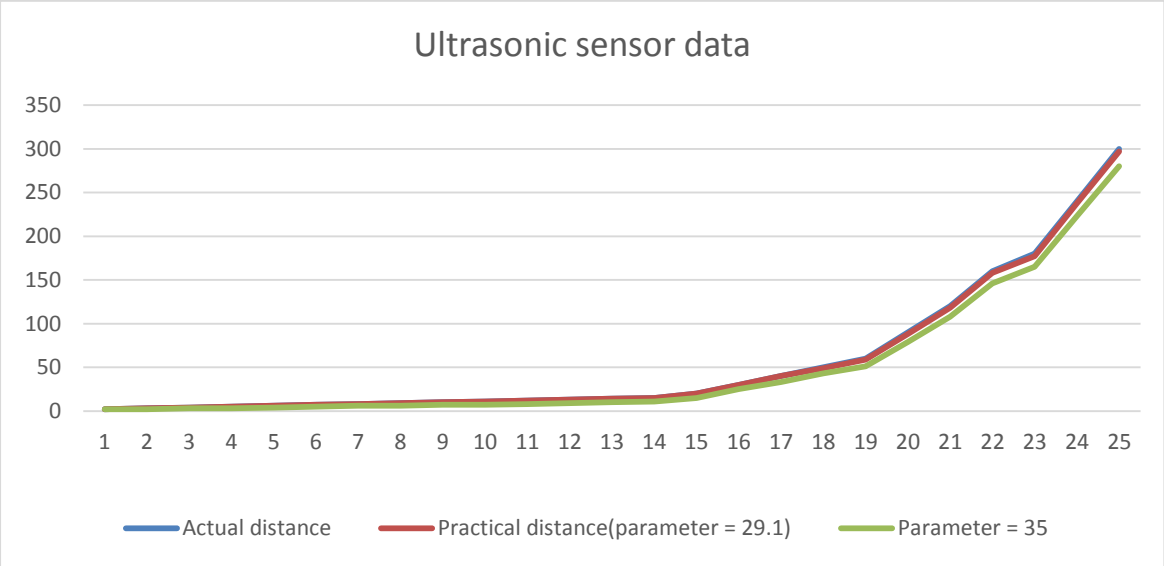
GND: GND

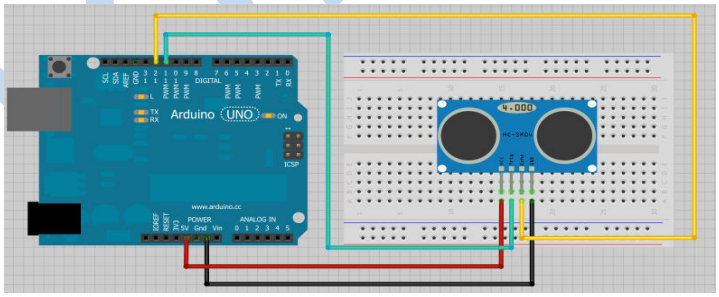
Actual distance	**Practical distance (parameter = 29.1)**	**Parameter = 35**
2	2	2
3	3	2
4	4	3
5	5	3
6	6	4
7	7	5
8	8	6
9	9	6
10	10	7
11	11	7
12	12	8
13	13	9
14	14	10
15	15	11
20	20	15
30	30	25
40	40	33
50	49	43
60	59	51
90	88	79
120	118	108

160	158	146
180	177	165
240	237.5	223
300	297	280

When Trigger pin was kept high for 5ms, range reduced to 2cm-235cm.

When Trigger pin was kept high for 15ms, range reduced to 2cm-235cm.





Connections with Arduino UNO

Source Code:

```
#define trigPin 7
```

```
#define echoPin 6


void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}


void loop() {
  long duration, distance;
  digitalWrite(trigPin, LOW);    // turn the LED off by making the voltage LOW
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);    // turn the LED off by making the voltage LOW
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);    // turn the LED off by making the voltage LOW
  duration = pulseIn(echoPin, HIGH);
  distance = (duration / 2) / 29.1;
  Serial.println(distance);
  if (distance < 4) {
    //Serial.println("object too close");
    digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(500);             // wait for a second
  }
  else {
    //Serial.println("cm");
```

```
    digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
    delay(500);
  }
  delay(500);
}
```

Applications:

- Vehicle detection
- Anti-collision detection in automated work environments
- For keeping a count of objects on the conveyer belt

Conclusion:

The ultrasonic distance sensor is a highly accurate sensor, especially if the obstacle is close to the sensor. It is an ideal sensor for IoT applications mentioned above.

The aim of this project was met and the distance of the obstacle from the sensor was to send an Android as serial data.

References:

[1] http://www.micropik.com/PDF/HCSR04.pdf
[2] https://www.optimusdigital.ro/index.php?controller=attachment&id_attachment=1

Experiment 8: Using OLED I2C Display with Arduino

Aim:

To use an OLED I2C display with Arduino.

Objective:

To learn about I2C protocol and using an I2C display with Arduino.

Components Required:

1. Arduino UNO
2. I2C 0.96" 128X64 White OLED Display
3. Jumper Wires

Connections:

Arduino Pins	OLED Display
5v	VCC
GND	GND
A4	SCL
A5	SDA

1. Perform the following connections as shown in the table.

Theory:

- The Inter-integrated Circuit (I2C) Protocol is a protocol intended to allow multiple “slave” digital integrated circuits (“chips”) to communicate with one or more “master” chips. Like the Serial Peripheral Interface (SPI), it is only intended for short distance communications within a single device. Like Asynchronous Serial Interfaces (such as RS-232 or UARTs), it only requires two signal wires to exchange information.

- There are two types of Display available: The SPI Display and I2C Display. The SPI is faster but uses more wires for connections and thus using up more pins of the Arduino. The I2C Display uses only 2 wires for communication.

Procedure:

- Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows us to display on the OLED screen.

- Go to link provided in Reference [1] and Reference [2] and download the libraries.
- Next in the Arduino IDE, click on:
- *Sketch >> Include Library >> Add .ZIP library.*
- Browse and select the folder where you downloaded the libraries.
- Now, we have to check if the OLED display is working fine. So, in your Arduino IDE click on:
- *File >> Examples >> Adafruit SSD1306 >> ssd1306_128x64_i2c.*
- Plug in your Arduino with the connections made to OLED display and upload this code. Open the Serial Monitor for debugging purposes.
- *Debugging:*
    - *If you receive an error while compiling stating Height incorrect, perform the following task:*

      Got to Your PC >> Documents >> Arduino >> libraries >> Adafruit SSD1306 >> Adafruit ssd1306.h >> Edit with notepad.

      Uncomment the line: #define SSD1306_128_64

      And comment the line : #define SSD1306_128_32

      This will fix the height error bug.
    - *If you receive an error stating no slave found at the address, that may be because your OLED Display may have a different slave address. In order to find out, with the connections made to OLED display, upload the code given below and Check your slave address on Serial monitor.*
- Change the address and output displayed on OLED Screen.

Source Code:

```
// --------------------------------------
// i2c_scanner
//
// Version 1
//    This program (or code that looks like it)
//    can be found in many places.
//    For example on the Arduino.cc forum.
//    The original author is not know.
// Version 2, Juni 2012, Using Arduino 1.0.1
//     Adapted to be as simple as possible by Arduino.cc user Krodal
// Version 3, Feb 26  2013
//    V3 by louarnold
// Version 4, March 3, 2013, Using Arduino 1.0.3
//    by Arduino.cc user Krodal.
//    Changes by louarnold removed.
//    Scanning addresses changed from 0...127 to 1...119,
```

```
//    according to the i2c scanner by Nick Gammon
//    http://www.gammon.com.au/forum/?id=10896
// Version 5, March 28, 2013
//    As version 4, but address scans now to 127.
//    A sensor seems to use address 120.
// Version 6, November 27, 2015.
//    Added waiting for the Leonardo serial communication.
//
//
// This sketch tests the standard 7-bit addresses
// Devices with higher bit address might not be seen properly.
//

#include <Wire.h>


void setup()
{
  Wire.begin();

  Serial.begin(9600);
  while (!Serial);             // Leonardo: wait for serial monitor
  Serial.println("\nI2C Scanner");
}


void loop()
{
  byte error, address;
  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {
    // The i2c_scanner uses the return value of
    // the Write.endTransmisstion to see if
    // a device did acknowledge to the address.
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Unknow error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
```

```
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000);           // wait 5 seconds for next scan
}
```

Applications:

- Display screen to view the sensor data in any system, using Arduino.

Conclusion:

Communication between OLED Display and Arduino is established and output viewed.

References:

[1] https://github.com/adafruit/Adafruit-GFX-Library

[2] https://github.com/adafruit/Adafruit_SSD1306

[3] http://playground.arduino.cc/Main/I2cScanner

[4] https://www.youtube.com/watch?v=FiPt7foZoAI

[5] https://learn.sparkfun.com/tutorials/i2c

Experiment 9: Digital clock using Arduino Pro Mini and Tiny RTC I2C OLED module.

Aim:

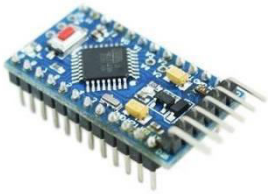
To develop a digital clock using RTC and Arduino Pro Mini.

Objective:

To learn about I2C protocol and using an I2C RTC Module with Arduino Pro Mini and develop a stand-alone digital clock.

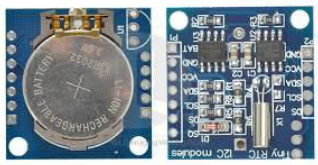
Components Required:

1. Arduino Pro Mini
2. I2C 0.96" 128X64 White OLED Display
3. Tiny RTC I2C Module
4. FTDI Basic Breakout
5. Jumper Wires
6. Mini B USB Connector



Connections:

Arduino Pro Mini	RTC Module
	VCC
GND	GND
A4	SDA
A5	SCL



RTC Module	OLED Display
VCC	VCC
GND	GND
SDA	SDA
SCL	SCL

Arduino Pro Mini	FTDI
5v	VCC

GND	GND
Tx	Rx
Rx	Tx

1. Perform the following connections as shown in the table.

   Note:
   - *Make sure you check the operating voltages of the Arduino Pro Mini and FTDI Basic Breakout for 5v/3v and power them accordingly.*
   - *Don't forget to short all the grounds.*
   - *Also the RTC has 2 connections for pin. Use one set with OLED and other set with Arduino Pro Mini.*

Theory:

- The Inter-integrated Circuit (I2C) Protocol is a protocol intended to allow multiple "slave" digital integrated circuits ("chips") to communicate with one or more "master" chips. Like the Serial Peripheral Interface (SPI), it is only intended for short distance communications within a single device. Like Asynchronous Serial Interfaces (such as RS-232 or UARTs), it only requires two signal wires to exchange information.

- There are two types of Display available: The SPI Display and I2C Display. The SPI is faster but uses more wires for connections and thus using up more pins of the Arduino. The I2C Display uses only 2 wires for communication.

- The FTDI Basic Breakout Board is used to program the Arduino Pro Mini. There are various models of USB to serial IC available.

Procedure:

- Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows us to display on the OLED screen.
- Go to link provided in Reference [1] and Reference [2] and Reference [3] and download the libraries.
- Next in the Arduino IDE, click on:
- *Sketch >> Include Library >> Add .ZIP library.*
- Browse and select the folder where you downloaded the libraries.
- Plug in your Arduino Pro Mini with the connections made as shown in the table above. Select the Board Arduino Pro Mini and Processor specific to your Board. Upload the code given below.

Note: You have to press the reset button every time you want to upload code to the Arduino Pro Mini Board. The on board LED on the FTDI Module blinks once in order to indicate when you should hit the reset button.

- Open the Serial Monitor for debugging purposes.
- If the OLED Display is not working then follow the steps:
- *File >> Examples >> Adafruit SSD1306 >> ssd1306_128x64_i2c.*
- *Debugging:*
  - *If you receive an error while compiling stating Height incorrect, perform the following task:*

    Got to Your PC >> Documents >> Arduino >> libraries >> Adafruit SSD1306 >> Adafruit ssd1306.h >> Edit with notepad.

    Uncomment the line: #define SSD1306_128_64

    And comment the line: #define SSD1306_128_32

    This will fix the height error bug.
  - *If you receive an error stating no slave found at the address, that may be because your OLED Display may have a different slave address. In order to find out, with the connections made to OLED display, upload the code given in Reference [4] and Check your slave address on Serial monitor.*
  - *Change the address and output displayed on OLED Screen.*

Source Code:

```
/**
 * Digital clock using RTC I2C module.
 * Code developed by : Ms. Michelle D'Souza, IoT Intern, 2016.
 * Grateful to the whole open source community on internet.
*/

//#include <SPI.h>
//#include <Wire.h>
//#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "RTClib.h"
#define OLED_RESET 4

Adafruit_SSD1306 display(OLED_RESET);
RTC_DS1307 rtc;

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday"};
//#if (SSD1306_LCDHEIGHT != 64)
//#error("Height incorrect, please fix Adafruit_SSD1306.h!");
//#endif
```

```
void setup () {
  //while (!Serial); // for Leonardo/Micro/Zero
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C
addr 0x3D (for the 128x64)
  display.clearDisplay();

  Serial.begin(9600);
  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);
  }

  if (! rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date & time this sketch was
compiled
     rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example to
set
    // January 21, 2014 at 3am you would call:
    // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
  }
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("Digital Clock");
  display.println();
  display.display();
  delay(2000);
  display.println("Loading . . .!");
  display.display();
  delay(2000);

}

void loop () {
  DateTime now = rtc.now();
  Serial.print(now.year(), DEC);
  Serial.print('/');
  Serial.print(now.month(), DEC);
  Serial.print('/');
  Serial.print(now.day(), DEC);
  Serial.print(" (");
  Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
  Serial.print(") ");
  Serial.print(now.hour(), DEC);
  Serial.print(':');
  Serial.print(now.minute(), DEC);
  Serial.print(':');
  Serial.print(now.second(), DEC);
  Serial.println();

  Serial.println();

  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(WHITE);
```

```
  display.setCursor(0, 0);
  display.println("Today:");
  display.print(now.year(), DEC);
  display.print('/');
  display.print(now.month(), DEC);
  display.print('/');
  display.print(now.day(), DEC);
  display.print(" (");
  display.print(daysOfTheWeek[now.dayOfTheWeek()]);
  display.print(") ");
  display.println();

  display.println("Time :");
  display.print(now.hour(), DEC);
  display.print(':');
  display.print(now.minute(), DEC);
  display.print(':');
  display.display();
  display.print(now.second(), DEC);
  display.display();
  delay(1000);
}
```

Applications:

Wrist watch.

Conclusion:

Digital clock is displayed on OLED Display using RTC Clock module and Arduino Pro Mini.

References:

[1] https://github.com/adafruit/Adafruit-GFX-Library

[2] https://github.com/adafruit/Adafruit_SSD1306

[3] https://github.com/adafruit/RTClib

[4] http://playground.arduino.cc/Main/I2cScanner

[5] https://learn.adafruit.com/ds1307-real-time-clock-breakout-board-kit/understanding-the-code

Experiment 10: Displaying custom images on OLED screen using Arduino Pro Mini

Aim:

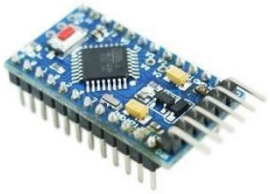
To display custom images on OLED screen using Arduino Pro Mini.

Objective:

To learn about I2C protocol and using an OLED Screen with Arduino Pro Mini to display custom monochrome images on it.

Components Required:

1. Arduino Pro Mini
2. I2C 0.96" 128X64 White OLED Display
3. FTDI Basic Breakout
5. Jumper Wires
6. Mini B USB Connector



Connections:

Arduino Pro Mini	OLED Display
5v	VCC
GND	GND
SDA	SDA
SCL	SCL

Arduino Pro Mini	FTDI
5v	VCC
GND	GND
Tx	Rx
Rx	Tx

2. Perform the following connections as shown in the table.

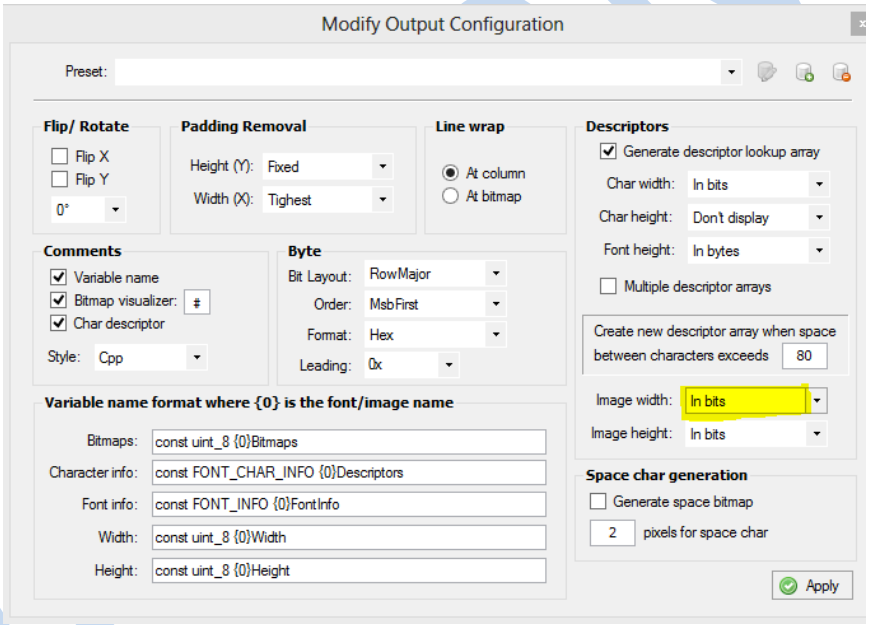
Note:

- *Make sure you check the operating voltages of the Arduino Pro Mini and FTDI Basic Breakout for 5v/3v and power them accordingly.*
- *Don't forget to short all the grounds.*
- *Also the RTC has 2 connections for pin. Use one set with OLED and other set with Arduino Pro Mini.*

Theory:

- The Inter-integrated Circuit (I2C) Protocol is a protocol intended to allow multiple "slave" digital integrated circuits ("chips") to communicate with one or more "master" chips. Like the Serial Peripheral Interface (SPI), it is only intended for short distance communications within a single device. Like Asynchronous Serial Interfaces (such as RS-232 or UARTs), it only requires two signal wires to exchange information.

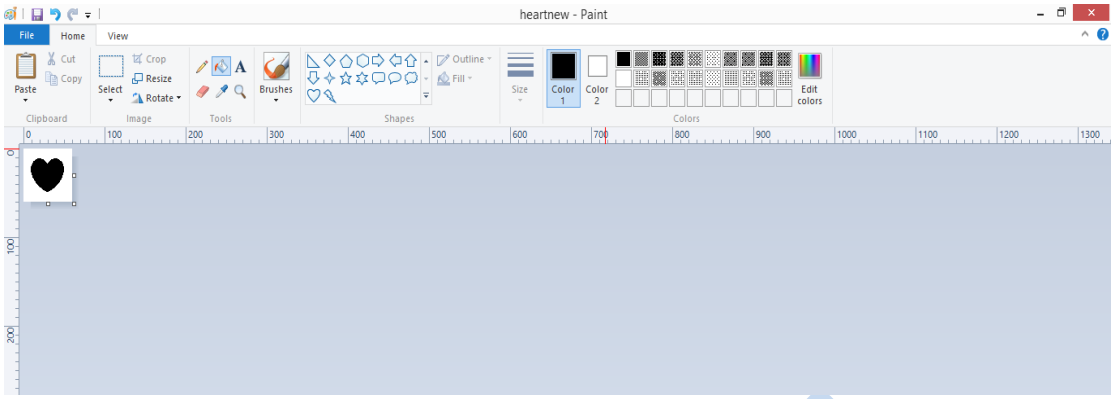
- There are two types of Display available: The SPI Display and I2C Display. The SPI is faster but uses more wires for connections and thus using up more pins of the Arduino. The I2C Display uses only 2 wires for communication.



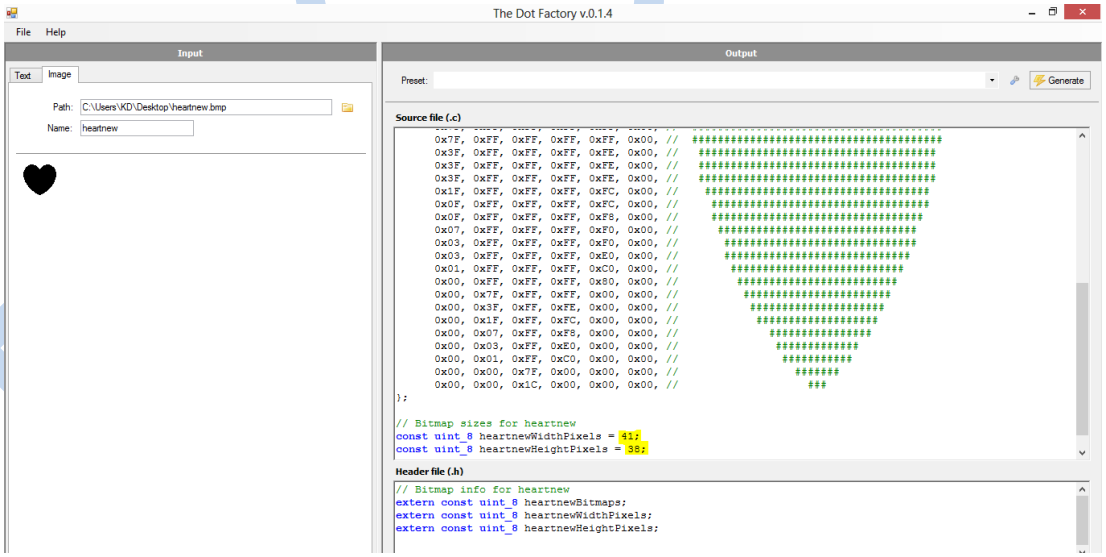
- The FTDI Basic Breakout Board is used to program the Arduino Pro Mini. There are various models of USB to serial IC available.

Procedure:

- In order to display custom images on OLED screen, we need to convert the image into a series of array containing the information.
- Let's begin with displaying a simple heart image so that we can use it in our next heart measurement system on OLED screen. Open up Paint application and first set the size. *Click Resize >> Pixels >> horizontal and vertical 60 x 60.*
- Select a simple Heart shape and fill it with black color as shown in image



- Next save it as Monochrome Bitmap type *>> heart.bmp*.
- Next, in order to convert this image to hexadecimal information, let's use the DOT Factory application. Go to Reference [1] and download and install the DOT Factory.
- Open the Dot Factory and Click *Image >> select heart.bmp >> settings (Next to Generate) >> Change Image width to In bits >> Apply*
- Now Click on generate and information should be displayed. This is the array which we require to copy paste in our code.
- Also take note of the pixel height and width. In the image shown below, its 41 and 38.



- Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows us to display on the OLED screen.
- Go to link provided in Reference [2] and download the library.
- Next in the Arduino IDE, click on:
- *Sketch >> Include Library >> Add .ZIP library.*
- Browse and select the folder where you downloaded the library.

- Next copy paste the code given below. You have to modify the contents of the code. Change the array contents to your array.
- Also change the width and height.
- Plug in your Arduino Pro Mini with the connections made as shown in the table above. Select the Board Arduino Pro Mini and Processor specific to your Board. Upload the code given below.

Note: You have to press the reset button every time you want to upload code to the Arduino Pro Mini Board. The on board LED on the FTDI Module blinks once in order to indicate when you should hit the reset button.

Source Code:

```
/**
   Displaying custom images on OLED screen.
   Code modified by : Ms. Michelle D'Souza, IoT Intern, 2016.
   Grateful to the whole open source community on internet.
*/

#include <Adafruit_SSD1306.h>

#define OLED_RESET 4//4 for pro mini
Adafruit_SSD1306 display(OLED_RESET);


static const unsigned char PROGMEM heart[] = {

  0x00, 0xF8, 0x00, 0x0F, 0x80, 0x00, //          #####                 #####
  0x03, 0xFF, 0x00, 0x7F, 0xE0, 0x00,
//       ##########          ##########
  0x07, 0xFF, 0x80, 0xFF, 0xF0, 0x00,
//      #############        #############
  0x1F, 0xFF, 0xC1, 0xFF, 0xFC, 0x00,
//     ################     ################
  0x1F, 0xFF, 0xE3, 0xFF, 0xFE, 0x00,
//     ################   ################
  0x3F, 0xFF, 0xF7, 0xFF, 0xFE, 0x00, //    ##################
##################
  0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0x00,
//    ######################################
  0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0x00,
//   ######################################
  0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0x00,
//   ######################################
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, //
########################################
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, //
########################################
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, //
########################################
```

```
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, //
########################################
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, //
########################################
```

```
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, //
########################################
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, //
########################################
  0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0x00,
//  #######################################
  0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0x00,
//  #######################################
  0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0x00,
//  #######################################
  0x3F, 0xFF, 0xFF, 0xFF, 0xFE, 0x00,
//   ####################################
  0x3F, 0xFF, 0xFF, 0xFF, 0xFE, 0x00,
//   ####################################
  0x3F, 0xFF, 0xFF, 0xFF, 0xFE, 0x00,
//   ####################################
  0x1F, 0xFF, 0xFF, 0xFF, 0xFC, 0x00,
//    ###################################
  0x0F, 0xFF, 0xFF, 0xFF, 0xFC, 0x00,
//     ##################################
  0x0F, 0xFF, 0xFF, 0xFF, 0xF8, 0x00,
//     ##################################
  0x07, 0xFF, 0xFF, 0xFF, 0xF0, 0x00,
//      #################################
  0x03, 0xFF, 0xFF, 0xFF, 0xF0, 0x00,
//       ################################
  0x03, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
//       ################################
  0x01, 0xFF, 0xFF, 0xFF, 0xC0, 0x00, //        ############################
  0x00, 0xFF, 0xFF, 0xFF, 0x80, 0x00, //         ##########################
  0x00, 0x7F, 0xFF, 0xFF, 0x00, 0x00, //          ########################
  0x00, 0x3F, 0xFF, 0xFE, 0x00, 0x00, //           ######################
  0x00, 0x1F, 0xFF, 0xFC, 0x00, 0x00, //            ####################
  0x00, 0x07, 0xFF, 0xF8, 0x00, 0x00, //              ################
  0x00, 0x03, 0xFF, 0xE0, 0x00, 0x00, //               #############
  0x00, 0x01, 0xFF, 0xC0, 0x00, 0x00, //                ###########
  0x00, 0x00, 0x7F, 0x00, 0x00, 0x00, //                  #######
  0x00, 0x00, 0x1C, 0x00, 0x00, 0x00, //                    ###
};


void setup() {
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C
addr 0x3D (for the 128x64)
  display.clearDisplay();
}

int BPM = 111;

//  Where the Magic Happens
void loop() {
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.print("BPM MEASUREMENT");
```

```
  //Change your height and width here
  display.drawBitmap(15, 20, heart, 41, 38, 1);
  display.setTextSize(3);
  display.setCursor(70, 30);
  //  display.println();
  display.print(BPM);
  display.display();
}
```

Applications:

Display custom images according to your design.

Conclusion:

Digital clock is displayed on OLED Display using RTC Clock module and Arduino Pro Mini.

]

References:

[1] http://www.eran.io/the-dot-factory-an-lcd-font-and-image-generator/

[2] https://github.com/adafruit/Adafruit_SSD1306

[3] http://www.nutsvolts.com/blog/post/building_your_own_bitmaps_and_animation_for_the_128x64_graphics_kit

[4] https://learn.adafruit.com/adafruit-gfx-graphics-library/graphics-primitives

Experiment 11 : Displaying Heart Beat Measurement on OLED Screen using Arduino Pro Mini

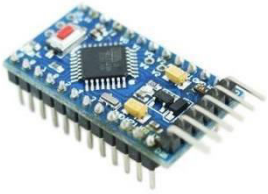
Aim:

To display heart beat on OLED screen using Pulse sensor and Arduino.

Objective:

To learn about I2C protocol and interfacing a pulse sensor with Arduino Pro Mini and develop a stand-alone heart beat measurement system.

Components Required:



1. Arduino Pro Mini
2. I2C 0.96" 128X64 White OLED Display
3. Pulse Sensor
4. FTDI Basic Breakout
5. Jumper Wires
6. Mini B USB Connector

Connections:

Arduino Pro Mini	Pulse Sensor
5v P.S	VCC
GND	GND
A0	SIG

Arduino Pro Mini	OLED Display
5v P.S	VCC
GND	GND
A4	SDA
A5	SCL

Arduino Pro Mini	FTDI
5v P.S	VCC
GND	GND
Tx	Rx
Rx	Tx

3. Perform the following connections as shown in the table.

   Note:
   - *Make sure you check the operating voltages of the Arduino Pro Mini and FTDI Basic Breakout for 5v/3v and power them accordingly.*
   - *Don't forget to short all the grounds.*
   - *Also the RTC has 2 connections for pin. Use one set with OLED and other set with Arduino Pro Mini.*

Theory:

- The Inter-integrated Circuit (I2C) Protocol is a protocol intended to allow multiple "slave" digital integrated circuits ("chips") to communicate with one or more "master" chips. Like the Serial Peripheral Interface (SPI), it is only intended for short distance communications within a single device. Like Asynchronous Serial Interfaces (such as RS-232 or UARTs), it only requires two signal wires to exchange information.

- There are two types of Display available: The SPI Display and I2C Display. The SPI is faster but uses more wires for connections and thus using up more pins of the Arduino. The I2C Display uses only 2 wires for communication.

- The FTDI Basic Breakout Board is used to program the Arduino Pro Mini. There are various models of USB to serial IC available.

Procedure:

- Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows us to display on the OLED screen.
- Go to link provided in Reference [1] and Reference [2] and download the libraries.
- Next in the Arduino IDE, click on:
- *Sketch >> Include Library >> Add .ZIP library.*
- Browse and select the folder where you downloaded the libraries.
- Next go to *Examples >> PulseSensor_Amped_Arduino-master >> PulseSensorAmped_Arduino-1dot4* and copy paste the code given below .Keep all the remaining files included.

- Plug in your Arduino Pro Mini with the connections made as shown in the table above. Select the Board Arduino Pro Mini and Processor specific to your Board. Upload the code given below.

Note: You have to press the reset button every time you want to upload code to the Arduino Pro Mini Board. The on board LED on the FTDI Module blinks once in order to indicate when you should hit the reset button.

- Open the Serial Monitor for debugging purposes.
- If the OLED Display is not working then follow the steps:
- *File >> Examples >> Adafruit SSD1306 >> ssd1306_128x64_i2c.*
- *Debugging:*
  - *If you receive an error while compiling stating Height incorrect, perform the following task:*
    Got to Your PC >> Documents >> Arduino >> libraries >> Adafruit SSD1306 >> Adafruit ssd1306.h >> Edit with notepad.
    Uncomment the line: #define SSD1306_128_64
    And comment the line: #define SSD1306_128_32
    This will fix the height error bug.
  - *If you receive an error stating no slave found at the address, that may be because your OLED Display may have a different slave address. In order to find out, with the connections made to OLED display, upload the code given in Reference[4] and Check your slave address on Serial monitor.*
  - *Change the address and output displayed on OLED Screen.*

Source Code:

```
/*

  Code modified by : Ms. Michelle D'Souza, IoT Intern, 2017.
   Grateful to the whole open source community on internet.

Pulse Sensor Amped 1.4    by Joel Murphy and Yury
Gitman   http://www.pulsesensor.com

  ----------------------  Notes ----------------------  ----------------------
---
  This code:
  1) Blinks an LED to User's Live Heartbeat   PIN 13
  2) Fades an LED to User's Live HeartBeat
  3) Determines BPM
  4) Prints All of the Above to Serial
```

```
  Read Me:
  https://github.com/WorldFamousElectronics/PulseSensor_Amped_Arduino/blob/
master/README.md
  ----------------------       ----------------------  -------------------
--
*/
#include <Adafruit_SSD1306.h>

#define OLED_RESET 4//4 for pro mini
Adafruit_SSD1306 display(OLED_RESET);


//  Variables
int pulsePin = 0;                 // Pulse Sensor purple wire connected to
analog pin 0
int blinkPin = 13;                // pin to blink led at each beat
int fadePin = 5;                  // pin to do fancy classy fading blink at
each beat
int fadeRate = 0;                 // used to fade LED on with PWM on
fadePin

// Volatile Variables, used in the interrupt service routine!
volatile int BPM;                   // int that holds raw Analog in 0.
updated every 2mS
volatile int Signal;                // holds the incoming raw data
volatile int IBI = 600;             // int that holds the time interval
between beats! Must be seeded!
volatile boolean Pulse = false;     // "True" when User's live heartbeat is
detected. "False" when not a "live beat".
volatile boolean QS = false;        // becomes true when Arduoino finds a
beat.

// Regards Serial OutPut  -- Set This Up to your needs
static boolean serialVisual = false;   // Set to 'false' by Default.  Re-
set to 'true' to see Arduino Serial Monitor ASCII Visual Pulse


static const unsigned char PROGMEM heart[] = {

  0x00, 0x01, 0xFF, 0x80, 0x00, 0xFF, 0xC0, 0x00, 0x00,
//               ##########               ##########
  0x00, 0x0F, 0xFF, 0xE0, 0x03, 0xFF, 0xF8, 0x00, 0x00,
//             ###############           ###############
  0x00, 0x1F, 0xFF, 0xF8, 0x0F, 0xFF, 0xFC, 0x00, 0x00,
//            ##################       ##################
  0x00, 0x3F, 0xFF, 0xFC, 0x1F, 0xFF, 0xFE, 0x00, 0x00,
//           ####################     ####################
  0x00, 0x7F, 0xFF, 0xFE, 0x3F, 0xFF, 0xFF, 0x00, 0x00,
//          ######################   ######################
  0x00, 0xFF, 0xFF, 0xFF, 0x7F, 0xFF, 0xFF, 0x80, 0x00,
//         ######################### #########################
  0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0x00,
//        ###################################################
  0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0x00,
//        ###################################################
  0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
//       #####################################################
```

```
  0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
//       #####################################################
  0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
//       #####################################################
  0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
//       #####################################################
  0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
//       #####################################################
  0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
//       #####################################################
  0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
//       #####################################################
  0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
//       #####################################################
  0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
//       #####################################################
  0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
//       #####################################################
  0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
//       #####################################################
  0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
//       #####################################################
  0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0x00,
//        ###################################################
  0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0x00,
//        ###################################################
  0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0x00,
//        ###################################################
  0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, 0x00,
//         #################################################
  0x00, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00,
//          ###############################################
  0x00, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0x00, 0x00,
//           #############################################
  0x00, 0x1F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0x00, 0x00,
//            ###########################################
  0x00, 0x1F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0x00, 0x00,
//            ###########################################
  0x00, 0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00,
//             #########################################
  0x00, 0x07, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00,
//              #######################################
  0x00, 0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00, 0x00,
//               #####################################
  0x00, 0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0x00, 0x00,
//                ###################################
  0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, 0x00, 0x00,
//                 #################################
  0x00, 0x00, 0x3F, 0xFF, 0xFF, 0xFE, 0x00, 0x00, 0x00,
//                   #############################
  0x00, 0x00, 0x1F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x00,
//                    ###########################
  0x00, 0x00, 0x0F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x00,
//                     #########################
  0x00, 0x00, 0x07, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00,
//                      #######################
  0x00, 0x00, 0x01, 0xFF, 0xFF, 0xC0, 0x00, 0x00, 0x00,
//                        ###################
```

```
  0x00, 0x00, 0x00, 0xFF, 0xFF, 0x80, 0x00, 0x00, 0x00,
//                          ################
  0x00, 0x00, 0x00, 0x7F, 0xFF, 0x00, 0x00, 0x00, 0x00,
//                           ###############
  0x00, 0x00, 0x00, 0x7F, 0xFF, 0x00, 0x00, 0x00, 0x00,
//                           ###############
  0x00, 0x00, 0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00,
//                             ###########
  0x00, 0x00, 0x00, 0x0F, 0xF8, 0x00, 0x00, 0x00, 0x00,
//                              #########
  0x00, 0x00, 0x00, 0x07, 0xF0, 0x00, 0x00, 0x00, 0x00,
//                               #######
  0x00, 0x00, 0x00, 0x03, 0xE0, 0x00, 0x00, 0x00, 0x00,
//                                #####
  0x00, 0x00, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x00, 0x00,
//                                  ###
};

void setup() {
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C
addr 0x3D (for the 128x64)
  display.clearDisplay();
  display.drawBitmap(0, 0,  heart, 35, 35, WHITE);
  delay(2000);
  display.clearDisplay();

  pinMode(blinkPin, OUTPUT);        // pin that will blink to your
heartbeat!
  pinMode(fadePin, OUTPUT);         // pin that will fade to your
heartbeat!
  Serial.begin(115200);             // we agree to talk fast!
  interruptSetup();                 // sets up to read Pulse Sensor signal
every 2mS
  // IF YOU ARE POWERING The Pulse Sensor AT VOLTAGE LESS THAN THE BOARD
VOLTAGE,
  // UN-COMMENT THE NEXT LINE AND APPLY THAT VOLTAGE TO THE A-REF PIN
  //   analogReference(EXTERNAL);
}


//  Where the Magic Happens
void loop() {

  //    serialOutput() ;

  if (QS == true) {     // A Heartbeat Was Found
    // BPM and IBI have been Determined
    // Quantified Self "QS" true when arduino finds a heartbeat
    fadeRate = 255;         // Makes the LED Fade Effect Happen
    // Set 'fadeRate' Variable to 255 to fade LED with pulse
    serialOutputWhenBeatHappens();   // A Beat Happened, Output that to
serial.
    QS = false;                      // reset the Quantified Self flag for
next time
  }

  ledFadeToBeat();                      // Makes the LED Fade Effect Happen
  delay(20);                             //  take a break
}
```

```
void ledFadeToBeat() {
  fadeRate -= 15;                         //  set LED fade value
  fadeRate = constrain(fadeRate, 0, 255); //  keep LED fade value from
going into negative numbers!
  analogWrite(fadePin, fadeRate);         //  fade LED
}
```

Conclusion:

Heartbeat is displayed on OLED Screen using Arduino Pro Mini.

References:

[1] https://github.com/adafruit/Adafruit_SSD1306

[2] https://github.com/WorldFamousElectronics/PulseSensor_Amped_Arduino

[3] http://playground.arduino.cc/Main/I2cScanner

Experiment 12 : Real time triggered automatic fish feeder system using RTC and Arduino

Aim:

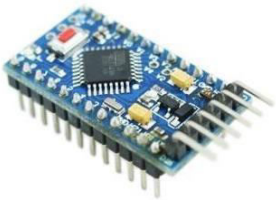
To develop real time triggered standalone fish feeder system.

Objective:

To learn about I2C protocol and using a Servo Motor and I2C RTC Module with Arduino Pro Mini and develop a stand-alone digital clock.

Components Required:

1. Arduino Pro Mini
2. I2C 0.96" 128X64 White OLED Display
3. Tiny RTC I2C Module
4. FTDI Basic Breakout
5. Jumper Wires
6. Mini B USB Connector
7. Servo Motor



Connections:

Arduino Pro Mini	RTC Module
5v P.S	VCC
GND	GND
A4	SDA
A5	SCL

RTC Module	OLED Display
VCC	VCC
GND	GND
SDA	SDA
SCL	SCL

Arduino Pro Mini	FTDI
5v P.S	VCC
GND	GND
Tx	Rx
Rx	Tx

Connect the Servo Motor to an external 5v Supply and not to the Pro Mini.

Arduino Pro Mini	Servo Motor
5v P.S	VCC
GND	GND
9	OUT

Perform the following connections as shown in the table.

Note:

- *Make sure you check the operating voltages of the Arduino Pro Mini and FTDI Basic Breakout for 5v/3v and power them accordingly.*
- *Don't forget to short all the grounds.*
- *Also the RTC has 2 connections for pin. Use one set with OLED and other set with Arduino Pro Mini.*

Theory:

- The Inter-integrated Circuit (I2C) Protocol is a protocol intended to allow multiple "slave" digital integrated circuits ("chips") to communicate with one or more "master" chips. Like the Serial Peripheral Interface (SPI), it is only intended for short distance communications within a single device. Like Asynchronous Serial Interfaces (such as RS-232 or UARTs), it only requires two signal wires to exchange information.

- There are two types of Display available: The SPI Display and I2C Display. The SPI is faster but uses more wires for connections and thus using up more pins of the Arduino. The I2C Display uses only 2 wires for communication.

- The FTDI Basic Breakout Board is used to program the Arduino Pro Mini. There are various models of USB to serial IC available.

Procedure:

- Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows us to display on the OLED screen.
- Go to link provided in Reference [1] and Reference [2] and download the libraries.
- Next in the Arduino IDE, click on:
- *Sketch >> Include Library >> Add .ZIP library.*
- Browse and select the folder where you downloaded the libraries.

- Next copy paste the code given below and change the hour alarm time if required.
- Plug in your Arduino Pro Mini with the connections made as shown in the table above. Select the Board Arduino Pro Mini and Processor specific to your Board. Upload the code given below.

Note: You have to press the reset button every time you want to upload code to the Arduino Pro Mini Board. The on board LED on the FTDI Module blinks once in order to indicate when you should hit the reset button.

- Open the Serial Monitor for debugging purposes.
- If the OLED Display is not working then follow the steps:
- *File >> Examples >> Adafruit SSD1306 >> ssd1306_128x64_i2c.*
- *Debugging:*
    - *If you receive an error while compiling stating Height incorrect, perform the following task:*
      Got to Your PC >> Documents >> Arduino >> libraries >> Adafruit SSD1306 >> Adafruit ssd1306.h >> Edit with notepad.
      Uncomment the line: #define SSD1306_128_64
      And comment the line: #define SSD1306_128_32
      This will fix the height error bug.
    - *If you receive an error stating no slave found at the address, that may be because your OLED Display may have a different slave address. In order to find out, with the connections made to OLED display, upload the code given in Reference[4] and Check your slave address on Serial monitor.*
    - *Change the address and output displayed on OLED Screen.*

Source Code:

```
/**
   Real time triggered Automatic Fish Feeder System.
   Code developed by : Ms. Michelle D'Souza, IoT Intern, 2017.
   Grateful to the whole open source community on internet.
   https://github.com/adafruit/RTClib
   https://learn.adafruit.com/ds1307-real-time-clock-breakout-board-kit/understanding-the-code
   https://www.hackster.io/robotgeek-projects-team/using-a-real-time-clock-with-arduino-6b3896
*/

#include <Adafruit_SSD1306.h>
#include "RTClib.h"
#include <Servo.h>
```

```
//#include <SPI.h>
//#include <Wire.h>
//#include <Adafruit_GFX.h>
#define OLED_RESET 4//4 for pro mini & uno
#define LED 13

Adafruit_SSD1306 display(OLED_RESET);
RTC_DS1307 rtc;
Servo myservo;  // create servo object to control a servo

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

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday"};
//#if (SSD1306_LCDHEIGHT != 64)
//#error("Height incorrect, please fix Adafruit_SSD1306.h!");
//#endif

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

  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C
addr 0x3D (for the 128x64)
  display.clearDisplay();

  Serial.begin(9600);
  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);
  }

  if (! rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date & time this sketch was
compiled
    // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }

  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("Ready?!");
  display.println();
  display.display();
  delay(2000);
  display.println("Loading Awesomeness!");
  display.display();
  delay(2000);

}

void loop () {
  DateTime now = rtc.now();
  int dayvar = now.dayOfTheWeek();
  int hourvar = now.hour();
  int minvar = now.minute();
  int secvar = now.second();
  //Serial.print(dayvar);
```

```
Serial.print(now.year());
Serial.print('/');
Serial.print(now.month());
Serial.print('/');
Serial.print(now.day());
Serial.print(" (");
Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
Serial.print(") ");
Serial.print(now.hour());
Serial.print(':');
Serial.print(now.minute());
Serial.print(':');
Serial.print(now.second());
Serial.println();
//Serial.print(now.dayOfTheWeek(), DEC);

Serial.println();

display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0, 0);
display.println("Today:");
display.print(now.year(), DEC);
display.print('/');
display.print(now.month(), DEC);
display.print('/');
display.print(now.day(), DEC);
//display.print(now.dayOfTheWeek(),DEC);

display.print(" (");
display.print(daysOfTheWeek[now.dayOfTheWeek()]);
display.print(") ");
display.println();

display.println("Time :");
display.print(now.hour(), DEC);
display.print(':');
display.print(now.minute(), DEC);
display.print(':');
display.display();
display.println(now.second(), DEC);
display.display();
//delay(2000);
//display.clearDisplay();

/*Set your time triggered automatic event here*/
switch (dayvar)
{
  case 3: //on Wednesday
    // Serial.println("inside case");
    {
      switch (hourvar)
      {
        case 22: //at 22 o'clock
          //  Serial.println("inside case");
          switch (minvar)
          {
```

```
              case 9 : //at 9 minutes
                {
                  digitalWrite(LED, HIGH);
                  display.println("----Alarm!----");
                  display.display();
                  Serial.println("Inside Case");
                  feeder();
                  break;
                }

            }
        }
      }
    case 0: //Sunday
      {
        switch (hourvar)
        {
          case 12: //at 12 o'clock
            //  Serial.println("inside case");
            switch (minvar)
            {
              case 30: //at 30 minutes
                switch (secvar)
                {
                  case 0: //at 1 second
                    digitalWrite(LED, HIGH);   // turn the Relay on (HIGH
is the voltage level)
                    display.println("----Alarm!----");
                    display.display();
                    delay(5000);               // wait for 5 seconds
                    digitalWrite(LED, LOW);    // turn the Relay off by
making the voltage LOW
                    delay(100);                // wait for 5 seconds
                    display.clearDisplay();
                }
            }
        }
      }
  }


  /*Any event to do daily at 22hr 15min*/
  switch (hourvar)
  {
    case 22: //at 22 o'clock
      //  Serial.println("inside case");
      switch (minvar)
      {
        case 30: //at 30 minutes
          switch (secvar)
          {
            case 1: //at 1 second
              digitalWrite(LED, HIGH);   // turn the Relay on (HIGH is the
voltage level)
              display.println("----Alarm!----");
              display.display();
              delay(5000);               // wait for 5 seconds
              digitalWrite(LED, LOW);    // turn the Relay off by making
the voltage LOW
```

```
                delay(100);                 // wait for 5 seconds
                display.clearDisplay();
            }
        }
    }
    delay(1000);
}

void feeder() {
  Serial.println("Inside Feeder");
  //Rotate the Servo motor to feed fish

  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180
degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in
variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach
the position
  }
  Serial.println("After 1st servo");

  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 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
  }

  digitalWrite(LED, LOW);    //
  display.clearDisplay();
  Serial.println("Leaving Feeder");

}
```

Applications:

Automatic plant watering system.

Conclusion:

Real time triggered events using RTC Clock module and Arduino Pro Mini are made possible.

References:

[1] https://github.com/adafruit/Adafruit_SSD1306

[2] https://github.com/adafruit/RTClib

[3] https://www.hackster.io/robotgeek-projects-team/using-a-real-time-clock-with-arduino-6b3896

[4] http://playground.arduino.cc/Main/I2cScanner

[5] https://learn.adafruit.com/ds1307-real-time-clock-breakout-board-kit/understanding-the-code

Experiment 13: Send Call Alert to mobile using GSM300 module if obstacle detected

Aim:

To create a simple Call alert system using GSM300 module and Arduino when obstacle detected.

Objective:

To learn about the various GSM300 commands and create a stand-alone Phone call alert system.

Components Required:

1. Arduino UNO
2. GSM300 Module
3. Jumper Wires
4. SIM 300

Connections:
Perform the following connections as shown in the table.

Arduino Pins	GSM300 Module
RX	9
TX	10

Following are some of the most frequently used AT commands:

COMMANDS	FUNCTION
AT+CGMI	To get the manufacturer's ID
AT+CMGF=1	Switch to text mode before sending SMS
AT+CMGS="+91xxxxxxxxxx"	Send SMS after typing the message & pressing **ctrl+z**
ATD999xxx1234	Dial to the mobile number provided
ATH	Disconnect the call

AT+CMSS	Send SMS from storage
AT+CMGR	Read one SMS at a time
AT+CMGD=1	Delete a Message.

Procedure:

1. Once you have your Adruino IDE installed and updated to the latest version, create a new sketch and copy paste the code contents provided below.
2. Change the +91xxxxxxxxx to a valid phone number to give the call alert.
3. Insert the SIM card in the GSM300 module and power it on.
4. The network led should blink once in 3 seconds after it has successfully found access to the network. It takes a few minutes to gain network access.

   Hint: You can give a call to the SIM Number inserted. If it connects, then the GSM300 module has gained network access!
5. Upload the program to the Arduino Board.
6. Make sure you have balance so that the Call is sent successfully.
7. Once the IR obstacle sensor detects the obstacle, a phone call alert is sent and the phone rings for 10 seconds before disconnecting.

Precaution: Power the GSM 300 Module before powering on the Arduino.

Source Code:

```
/**
   Code modified by : Ms.Michelle D'Souza, IoT Intern, 2016
   Thanks to:
   https://alselectro.wordpress.com/2013/02/07/arduino-with-gsm/
   http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-
input/arduino-ir-obstacle-sensor-tutorial-and-manual/
   http://www.circuitstoday.com/interface-gsm-module-with-arduino
**/
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9, 10);

int LED = 13; // Use the onboard Uno LED
int isObstaclePin = 3;  // This is our input pin
int isObstacle = HIGH;  // HIGH MEANS NO OBSTACLE

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(isObstaclePin, INPUT);
  mySerial.begin(9600);   // Setting the baud rate of GSM Module
  Serial.begin(9600);

}
```

```
void loop() {
  isObstacle = digitalRead(isObstaclePin);
  if (isObstacle == HIGH)
  {
    Serial.println("OBSTACLE!!, OBSTACLE!!");
    digitalWrite(LED, HIGH);
    CallAlert();
  }
  else
  {
    Serial.println("clear");
    digitalWrite(LED, LOW);
  }
  delay(200);
}


void CallAlert()
{
  Serial.println("AT"); // Wake up GSM
  mySerial.println("AT"); // Wake up GSM

  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  Serial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode

  mySerial.print("ATD"); //Dial the phone number using ATD command
  mySerial.print(xxxxxxxxxx);//Valid 10 digit phone number

  mySerial.println(";"); // Semicolon is a must at the end
  delay(10000);
  mySerial.println("ATH"); // After a delay of 10 secs Hang the call
  Serial.println("ATH"); // After a delay of 10 secs Hang the call

}
```

Applications:

- Call Alert for any system can be developed, if sensor value is above certain threshold value.

Conclusion:

With the use of GSM300 module and Arduino, easy Call alert system can be developed for any application.

References:

[1] http://www.circuitstoday.com/interface-gsm-module-with-arduino

[2] https://github.com/miguelbalboa/rfid http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/arduino-ir-obstacle-sensor-tutorial-and-manual/

[3] https://alselectro.wordpress.com/tag/gsm300-tutorial/

[4] http://probots.co.in/Manuals/SIM300.pdf

Experiment 14 : Send SMS Alert using GSM300 Module if Obstacle detected

Aim:

To create a simple SMS alert system using GSM300 module and Arduino when obstacle detected.

Objective:

To learn about the various GSM commands and create a stand-alone SMS alert system.

Components Required:

1. Arduino UNO
2. GSM300 Module
3. Jumper Wires
4. SIM 300

Connections:
Perform the following connections as shown in the table.

Arduino Pins	GSM300 Module
RX	9
TX	10

Following are some of the most frequently used AT commands :

COMMANDS	FUNCTION
AT+CGMI	To get the manufacturer's ID
AT+CMGF=1	Switch to text mode before sending SMS
AT+CMGS="+91xxxxxxxxx"	Send SMS after typing the message & pressing **ctrl+z**
ATD999xxx1234	Dial to the mobile number provided
ATH	Disconnect the call

AT+CMSS	Send SMS from storage
AT+CMGR	Read one SMS at a time
AT+CMGD=1	Delete a Message.

Procedure:

8. Once you have your Adruino IDE installed and updated to the latest version, create a new sketch and copy paste the code contents provided below.
9. Change the +91xxxxxxxxx to a valid phone number to view the sent SMS.
10. Insert the SIM card in the GSM300 module and power it on.
11. The network led should blink once in 3 seconds after it has successfully found access to the network. It takes a few minutes to gain network access.
    Hint : You can give a call to the SIM Number inserted. If it connects, then the GSM300 module has gained network access!
12. Upload the program to the Arduino Board.
13. Make sure you have balance so that the SMS is sent successfully.
14. Once the IR obstacle sensor detects the obstacle, an sms to the number is sent.

Precaution: Power the GSM 300 Module before powering on the Arduino.

Source Code:

```
/**
 * Code modified by : Ms.Michelle D'Souza, IoT Intern, 2016
 * Thanks to:
 * http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-
input/arduino-ir-obstacle-sensor-tutorial-and-manual/
 * http://www.circuitstoday.com/interface-gsm-module-with-arduino
**/
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9, 10);

int LED = 13; // Use the onboard Uno LED
int isObstaclePin = 3;  // This is our input pin
int isObstacle = HIGH;  // HIGH MEANS NO OBSTACLE

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(isObstaclePin, INPUT);
  mySerial.begin(9600);   // Setting the baud rate of GSM Module
  Serial.begin(9600);

}

void loop() {
```

```
  isObstacle = digitalRead(isObstaclePin);
  if (isObstacle == HIGH)
  {
    Serial.println("OBSTACLE!!, OBSTACLE!!");
    digitalWrite(LED, HIGH);
    SendMessage();
  }
  else
  {
    Serial.println("clear");
    digitalWrite(LED, LOW);
  }
  delay(200);
}


void SendMessage()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  Serial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+91xxxxxxxxxx\"\r"); // Replace x with mobile
number
  Serial.println("AT+CMGS=\"+91xxxxxxxxxx\"\r"); // Replace x with mobile
number
  delay(1000);
  mySerial.println("I am SMS from GSM Module!  Obstacle is detected!");//
The SMS text you want to send
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
  Serial.println("sent");
}
```

Applications:

- SMS Alert for any system can be developed, if sensor value above certain threshold value.

Conclusion:

With the use of GSM300 module and Arduino, easy SMS alert system can be developed for any applicaton.

References:

[1] http://www.circuitstoday.com/interface-gsm-module-with-arduino

[2] https://github.com/miguelbalboa/rfid http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/arduino-ir-obstacle-sensor-tutorial-and-manual/

[3] https://alselectro.wordpress.com/tag/gsm300-tutorial/

[4] http://probots.co.in/Manuals/SIM300.pdf

Experiment 15: Extract GPS Coordinates from GPS module using Arduino

Aim:

To view the GPS Coordinates from GPS module on serial monitor of Arduino.

Objective:

To learn about the various GPS commands and extract the latitude and longitude data using Arduino.

Components Required:

1. Arduino UNO
2. GPS Module (5v TTL)
3. Jumper Wires

Connections:
Perform the following connections as shown in the table.

Arduino Pins	GPS Module
RX	9
TX	10
VCC	5v
GND	GND

Procedure:

15. Once you have your Arduino IDE installed and updated to the latest version, create a new sketch and copy paste the code contents provided below.
16. Make the connections and upload the code. The blue power Led should turn on.
17. The GPS receiver will send raw GPS data based on NMEA 0183 protocol.
18. GPRMC header : GPRMC tells the latitude, longitude, speed, time and date. The details of GPRMC message is shown below.

$GPRMC,161229.487,A,3723.2475,N,12158.3416,W,0.13,309.62,120598,,*10

The information contained in the following line is shown in table below:

Table 1.9 – RMC Data Format

Name	Example	Units	Description
Message ID	$GPRMC		RMC protocol header
UTC position	161229.487		hhmmss.sss
Status	A		A=data valid or V data not valid
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W	W		E=east or W=west
Speed Over Ground	0.13	knots	
Course Over Ground	309.62	degrees	True
Date	120598		ddmmyy
Magnetic Variation		degrees	E=east or W=west
Checksum	*10		
<CR><LF>			End of message termination

19. We need to extract the latitude and longitude information from this Header.
20. The signal led should blink once in 3 seconds after it has successfully found access to the satellite.
21. View the Latitude and longitude on serial monitor.

Precaution: Power the GSM 300 Module before powering on the Arduino.

Source Code:

```
/*********************************************************************************
 Program to read GPS data and print data to serial monitor.
 BOARD   : Arduino Uno
 http://www.rhydolabz.com/wiki/?p=658
 *********************************************************************************/

int Gpsdata;          // for incoming serial data
unsigned int finish = 0; // indicate end of message
unsigned int pos_cnt = 0; // position counter
unsigned int lat_cnt = 0; // latitude data counter
unsigned int log_cnt = 0; // longitude data counter
unsigned int flg    = 0; // GPS flag
unsigned int com_cnt = 0; // comma counter
char lat[20];         // latitude array
char lg[20];          // longitude array

void Receive_GPS_Data();

/*********************************************************************************
 Function    : setup()
 Description : Use it to initialize variables, pin modes, start using libraries, etc.
          The setup function will only run once, after each power up or reset
          of the Arduino board.
*********************************************************************************/
```

```
void setup()
{
 Serial.begin(9600);   // opens serial port, sets data rate to 9600 bps
}

/*********************************************************************************
 Function    : loop()
 Description : loops consecutively, allowing your program to change and respond.
           Use it to actively control the Arduino board.
*********************************************************************************/
void loop()
{
 Receive_GPS_Data();
 Serial.print("Latitude  : ");
 Serial.println(lat);
 Serial.print("Longitude : ");
 Serial.println(lg);
 finish = 0; pos_cnt = 0;

}
/*********************************************************************************
 Function    : Receive_GPS_Data()
 Description : finding Latitudse and longitude from GPRMC message

*********************************************************************************/
void Receive_GPS_Data()
{
 while (finish == 0) {
  while (Serial.available() > 0) {    // Check GPS data
   Gpsdata = Serial.read();
   flg = 1;
   if ( Gpsdata == '$' && pos_cnt == 0) // finding GPRMC header
    pos_cnt = 1;
   if ( Gpsdata == 'G' && pos_cnt == 1)
    pos_cnt = 2;
   if ( Gpsdata == 'P' && pos_cnt == 2)
    pos_cnt = 3;
   if ( Gpsdata == 'R' && pos_cnt == 3)
    pos_cnt = 4;
   if ( Gpsdata == 'M' && pos_cnt == 4)
    pos_cnt = 5;
   if ( Gpsdata == 'C' && pos_cnt == 5 )
    pos_cnt = 6;
   if (pos_cnt == 6 &&  Gpsdata == ',') { // count commas in message
    com_cnt++;
    flg = 0;
   }

   if (com_cnt == 3 && flg == 1) {
    lat[lat_cnt++] =  Gpsdata;       // latitude
    flg = 0;
   }

   if (com_cnt == 5 && flg == 1) {
    lg[log_cnt++] =  Gpsdata;       // Longitude
```

```
      flg = 0;
    }

    if ( Gpsdata == '*' && com_cnt >= 5) {
      com_cnt = 0;                  // end of GPRMC message
      lat_cnt = 0;
      log_cnt = 0;
      flg     = 0;
      finish  = 1;

    }
   }
  }
}
```

Applications:

- Gps data can be sent from Arduino to the internet portals like Thingspeak and tracking system can be developed.

Conclusion:

With the use of GPS module and Arduino, the latitude and longitude coordinates were extracted and displayed.

References:

[1] http://www.rhydolabz.com/wiki/wp-content/uploads/gps/mini_gps_usermanual_a01.pdf

[2] http://www.rhydolabz.com/wiki/?p=658

Experiment 16 : Remote LED control over the internet using Arduino Yun and Thingspeak

Aim:

To control an LED remotely from the internet using Arduino Yun board.

Objective:

To learn to perform Thingspeak Talkback.

Components Required:

1. Arduino Yún board
2. Micro-B USB cable

Procedure:

I. Connect Arduino Yún to WiFi

First you need to power the Arduino Yún. Connect the Arduino Yún to your computer with a Micro-B USB cable. This USB cable provides power and data to the board.
Once powered up, the Arduino Yún will broadcast a Wi-Fi network called, "ArduinoYun-XXXXXXXXXXXX". Connect to this Wi-Fi Network using a laptop, smartphone, or tablet. Open a web browser and go to, the following site address: http://arduino.local or 192.168.240.1 – this will bring up the Arduino Yún Wi-Fi Configuration. Enter the password of "arduino" to start the setup. Click Configure. Under the Wireless Parameters section, enter your Wi-Fi network name (SSID), network encryption type, and password. This wi-Fi information is for the Wi-Fi network that you want the Arduino Yún to connect to. Once the information is entered, press "Configure & Restart" to have the Arduino Yún connect to the specified Wi-Fi network.

II. Setting up the Arduino Yún IDE

- The Arduino Yún is only supported by Arduino IDE 1.5.4 and later. Download the Arduino IDE 1.5.4 or newer at Arduino.cc.
- When programming the Arduino Yún, you must choose Arduino Yún from the Tools > Board menu in the Arduino IDE.

III. Create ThingSpeak User Account and Setup a New TalkBack

- To get started with ThingSpeak, sign up for a free user account at ThingSpeak.com by clicking Sign Up or Sign In if you already have a ThingSpeak account.

- Create a new ThingSpeak Channel by selecting Channels and “Create New Channel” – a Channel is where you are able to store data collected by the Arduino Yún.
- Create a new TalkBack by selecting Apps, TalkBack, and “New TalkBack” – a TalkBack is a ThingSpeak App that allows you to store commands that the Arduino Yún can process.

IV. Program the Arduino Yún with the TalkBack App

- Use the code given below
- Change the TalkBack API Key to the TalkBack API Key created in the previous step
- Change the TalkBack ID to the TalkBack ID created in the previous step
- Select Tools -> Board -> Arduino Yún
- Select Tools -> Port -> (Select the Arduino Yún COM Port)
- Upload the sketch to your Arduino Yún
- Open the Serial Monitor

V. Control the Arduino Yún with TalkBack

- Enter the command “ON” into the TalkBack
- Enter the command “OFF” into the TalkBack

The Arduino Yún will periodically check the TalkBack to see if a command is queued up. If the command is “ON” the Arduino’s on-board LED (on GPIO 13) will light up. If the command is “OFF” the Arduino’s on-board LED will turn off. The TalkBack commands can be entered on ThingSpeak or via the TalkBack API.

Source Code:

```
#include "Bridge.h"
 #include "HttpClient.h"

 //ThingSpeak Settings
 String thingSpeakAPI = "api.thingspeak.com";
 String talkBackAPIKey = "PLC52OLN09W7AE10";
 String talkBackID = "9323";
 const int checkTalkBackInterval = 15 * 1000;    // Time interval in milliseconds to check TalkBack
(number of seconds * 1000 = interval)
 // Variable Setup
 long lastConnectionTime = 0;

 void setup()
 {
   // Setup On-board LED
   pinMode(13, OUTPUT);
   digitalWrite(13, LOW);
   delay(1000);
   digitalWrite(13, HIGH);
   delay(1000);
   digitalWrite(13, LOW);
   // Initialize Bridge
```

```
  Bridge.begin();
  // Initialize Serial
  Serial.begin(9600);
  while(!Serial);
}

void loop()
{
  // Check ThingSpeak for TalkBack Commands
  checkTalkBack();
  delay(checkTalkBackInterval);
}

void checkTalkBack()
{
  HttpClient client;
  String talkBackCommand;
  char charIn;
  String talkBackURL =  "http://" + thingSpeakAPI + "/talkbacks/" + talkBackID +
"/commands/execute?api_key=" + talkBackAPIKey;
  // Make a HTTP GET request to the TalkBack API:
  client.get(talkBackURL);
  while (client.available()) {
    charIn = client.read();
    talkBackCommand += charIn;
  }
  // Turn On/Off the On-board LED
  if (talkBackCommand == "ON")
  {
    Serial.println(talkBackCommand);
    digitalWrite(13, HIGH);
  }
  else if (talkBackCommand == "OFF")
  {
    Serial.println(talkBackCommand);
    digitalWrite(13, LOW);
  }
  Serial.flush();
  delay(1000);
}
```

Applications:

- Many appliances can be controlled remotely over the internet.

Conclusion:

Using Thingspeak, a workable prototype of an IoT project was created. It can be concluded that Arduino, along with other sensors and modules, can create many successful IoT projects.

References:

[1] http://community.thingspeak.com/tutorials/arduino/controlling-the-arduino-yun-with-talkback/

Experiment 17: To send an SMS if obstacle is detected (Arduino Yun)

Aim:

To send SMS alert from Arduino Yun if obstacle is detected.

Objective:

To learn to send online SMS and integrate it with Arduino code.

Components Required:

1. Arduino Yun board
2. Ultrasonic sensor

Online requirements:

1. Temboo account
2. Twilio account

Connections:

Arduino Pins	Ultrasonic sensor Pins
5V	VCC
GND	GND
Pin 7	Trig
Pin 6	Echo

Procedure:

1. Make an account on Temboo and Twilio and note down the Account SID and Auth Token from your Twilio account.
2. Verify your phone number in the account and obtain a free number given by Twilio.
3. Make the connections of the Arduino Yun and the Ultrasonic sensor.
4. Make sure the Yun in connected to the internet.
5. Note down your Account Name, APP_NAME and APP_KEY of your Temboo account and replace those values in the header file.
6. Upload the code to the Arduino Yun and observe.

Source Code:

Temboo Header File:

```
#define TEMBOO_ACCOUNT "ACCOUNT_NAME"  // your Temboo account name
#define TEMBOO_APP_KEY_NAME "APP_NAME"  // your Temboo app key name
#define TEMBOO_APP_KEY  "APP_KEY"  // your Temboo app key
```

Main code:

```
#define trigPin 7
#define echoPin 6

#include <Bridge.h>
#include <Temboo.h>
#include "TembooAccount.h" // contains Temboo account information

// Note that for additional security and reusability, you could
// use #define statements to specify these values in a .h file.
long duration, distance;
// the Account SID from your Twilio account
const String TWILIO_ACCOUNT_SID = "";

// the Auth Token from your Twilio account
const String TWILIO_AUTH_TOKEN = "";

// your Twilio phone number, e.g., "+1 555-222-1212"
const String TWILIO_NUMBER = "";

// the number to which the SMS should be sent, e.g., "+1 555-222-1212"
const String RECIPIENT_NUMBER = “”;

boolean success = false;

void setup() {
  // put your setup code here, to run once:
  Bridge.begin();       // initialize Bridge
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  delay(4000);
  //  while(!Serial);
  //  Bridge.begin();
}

void loop() {
```

```
 success = false;
 ultra();
 if (distance < 4)
 {
  SMS();
 }
 delay(500);
}

void ultra()
{
 digitalWrite(trigPin, LOW);   // turn the LED off by making the voltage LOW
 delayMicroseconds(2);
 digitalWrite(trigPin, HIGH);   // turn the LED off by making the voltage LOW
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);   // turn the LED off by making the voltage LOW
 duration = pulseIn(echoPin, HIGH);
 distance = (duration / 2) / 29.1;
 Serial.println(distance);
}

void SMS()
{
   success = false;
   if (!success) {
    digitalWrite(13, HIGH);

    Serial.println("Running SendAnSMS...");

    // we need a Process object to send a Choreo request to Temboo
    TembooChoreo SendSMSChoreo;

    // invoke the Temboo client
    // NOTE that the client must be reinvoked and repopulated with
    // appropriate arguments each time its run() method is called.
    SendSMSChoreo.begin();

    // set Temboo account credentials
    SendSMSChoreo.setAccountName(TEMBOO_ACCOUNT);
    SendSMSChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
    SendSMSChoreo.setAppKey(TEMBOO_APP_KEY);

    // identify the Temboo Library choreo to run (Twilio > SMSMessages > SendSMS)
    SendSMSChoreo.setChoreo("/Library/Twilio/SMSMessages/SendSMS");
```

```
  // set the required choreo inputs
  // see https://www.temboo.com/library/Library/Twilio/SMSMessages/SendSMS/
  // for complete details about the inputs for this Choreo

  // the first input is a your AccountSID
  SendSMSChoreo.addInput("AccountSID", TWILIO_ACCOUNT_SID);

  // next is your Auth Token
  SendSMSChoreo.addInput("AuthToken", TWILIO_AUTH_TOKEN);

  // next is your Twilio phone number
  SendSMSChoreo.addInput("From", TWILIO_NUMBER);

  // next, what number to send the SMS to
  SendSMSChoreo.addInput("To", RECIPIENT_NUMBER);

  // finally, the text of the message to send
 SendSMSChoreo.addInput("Body", "Obstacle detected!");

  // tell the Process to run and wait for the results. The
  // return code (returnCode) will tell us whether the Temboo client
  // was able to send our request to the Temboo servers
  unsigned int returnCode = SendSMSChoreo.run();

  // a return code of zero (0) means everything worked
  if (returnCode == 0) {
    Serial.println("Success! SMS sent!");
    success = true;
  } else {
    // a non-zero return code means there was an error
    // read and print the error message
    while (SendSMSChoreo.available()) {
      char c = SendSMSChoreo.read();
      Serial.print(c);
    }
  }
  SendSMSChoreo.close();

  // do nothing for the next 10 seconds
  Serial.println("Waiting...");
  delay(10000);
}
digitalWrite(13, LOW);
```

```
  }
}
```

Applications:

- To receive alerts for security purposes.
- To check if a window is open, intruder alert etc.

Conclusion:

Using Temboo and Twilio, a workable prototype of an IoT project was created. It can be concluded that Arduino, along with other sensors and modules, can create many successful IoT projects.

References:

[1] https://temboo.com/arduino/yun/send-sms

Experiment 18 : Location of vehicle using Arduino and Tracking shield

Aim:

To create a vehicle tracking system using Arduino and tracking shield.

Objective:

To create a stand-alone system used to track the vehicle and see the position on Google Maps.

Components Required:

1. Arduino UNO
2. Tracking Shield - GSM/GPRS/GPS (SIM800)
3. Jumper Wires
4. SIM 300
5. Power supply

Connections:

Insert the Tracking shield on the Arduino as shown in the picture alongside.

Note: It's recommended to power the Arduino externally instead of the USB power since the Tracking Shield draws huge current

Procedure:

22. Once you have your Arduino IDE installed and updated to the latest version, create a new sketch and copy paste the code contents provided below.
23. Change the +91xxxxxxxxx in the code to a valid phone number which will receive the position of vehicle.
24. Upload the program to the Arduino Board.
25. Insert the SIM card in the Tracking shield module and connect the antenna of the GSM and GPS slots respectively as shown in the image above.
26. Insert the shield in the Arduino Board and power it on. The Power led should turn on.
27. The STS led indicates the operating status of the module. Power the Arduino board externally for proper functioning.
28. The 3D Led should turn off once GPS is fixed. Point the antenna towards open sky if led keeps blinking.
29. The network led should blink once in 3 seconds after it has successfully found access to the network. It takes a few minutes to gain network access.
    Hint: You can give a call to the SIM Number inserted. If it connects, then the Tracking Shield has gained network access!
30. Make sure you have balance so that the SMS is sent successfully.
31. Once a call is received from the intended sender, an SMS with an http link will be sent. Open the link and the location of the vehicle will be shown on Google maps. Track your vehicle whenever you want by just shooting a call!

Source Code:

```
/**
 * Code developed by : Ms. Michelle D'Souza, IoT Intern, 2016.
 * Grateful to the open source community on the Internet
 * Vehicle Tracking System.
 * SMS is sent if call is received from specified number
 */

#include <SoftwareSerial.h>
#include <SPI.h>

SoftwareSerial GPS(4, 5);                              /* Software serial -
RX, TX pins                          */

int smsflag = 0, sendflg = 0;
int Gpsdata;                 // for incoming serial data
unsigned int finish = 0; // indicate end of message
unsigned int pos_cnt = 0; // position counter
unsigned int lat_cnt = 0; // latitude data counter
unsigned int log_cnt = 0; // longitude data counter
unsigned int flg    = 0; // GPS flag
unsigned int com_cnt = 0; // comma counter
```

```
char lat[20];             // latitude array
char lg[20];              // longitude array
char Rec_data;

unsigned int lat_cntP = 0; // latitude data position counter
unsigned int log_cntP = 0; // longitude data position counter
char latP[20];             // latitude array position
char lgP[20];              // longitude array position

unsigned int AorVdata = 0; // latitude data position counter
char AorV[20];             // latitude array position

float degLat; //Will hold positin data in simple degree format
float degWholeLat; //Variable for the whole part of position
float degDecLat;  //Variable for the decimal part of degree

float degLong; //Will hold positin data in simple degree format
float degWholeLong; //Variable for the whole part of position
float degDecLong;  //Variable for the decimal part of degree

void setup()
{
  Serial.begin(9600);                                  /* Initialize serial
communication at 9600 bits per second */
  GPS.begin(9600);                                     /* Initialize software
serial at 9600 bps for GPS          */

  Serial.print("AT\r\n");                              /* Initialization
command                                  */
  delay(1000);
  Serial.print("ATE0\r\n");                            /* Turn echo
off                                                 */
  delay(1000);
  Serial.print("AT+CMGF=1\r\n");                       /* Text
mode                                              */
  delay(1000);
  Serial.print("AT+CNMI=2,1,0,0,0\r\n");               /* Set message
format                                        */
  delay(1000);
  //Serial.print("AT+CLIP=1\r\n");                     /* Set message
format                                        */
  //delay(1000);
}


void loop() {
  while (Serial.available())
  {
    String calling = Serial.readString();
    //Serial.print(received);
    String callerID = calling.substring(20, 30);
    Serial.println(callerID);
    if (callerID == "9876543210") {//change to the senders nmber
      Serial.print("ATH\r\n");
      Receive_GPS_Data();
      Serial.println("back inside main -resetting");
      finish = 0; pos_cnt = 0;

    }
```

```
  }
}

void Receive_GPS_Data()
{

  while (finish == 0) {
    while (GPS.available() > 0) {      // Check GPS data
      Gpsdata = GPS.read();
      flg = 1;
      if ( Gpsdata == '$' && pos_cnt == 0) // finding GPRMC header
        pos_cnt = 1;
      if ( Gpsdata == 'G' && pos_cnt == 1)
        pos_cnt = 2;
      if ( Gpsdata == 'P' && pos_cnt == 2)
        pos_cnt = 3;
      if ( Gpsdata == 'R' && pos_cnt == 3)
        pos_cnt = 4;
      if ( Gpsdata == 'M' && pos_cnt == 4)
        pos_cnt = 5;
      if ( Gpsdata == 'C' && pos_cnt == 5 )
        pos_cnt = 6;
      if (pos_cnt == 6 &&  Gpsdata == ',') { // count commas in message
        com_cnt++;
        flg = 0;
      }

      if (com_cnt == 2 && flg == 1) {
        AorV[AorVdata++] =  Gpsdata;          // latitude
        flg = 0;
      }


      if (com_cnt == 3 && flg == 1) {
        lat[lat_cnt++] =  Gpsdata;          // latitude
        flg = 0;
      }


      if (com_cnt == 4 && flg == 1) {
        latP[lat_cntP++] =  Gpsdata;          // latitude
        flg = 0;
      }

      if (com_cnt == 5 && flg == 1) {
        lg[log_cnt++] =  Gpsdata;          // Longitude
        flg = 0;
      }


      if (com_cnt == 6 && flg == 1) {
        lgP[log_cntP++] =  Gpsdata;          // Longitude
        flg = 0;
      }

      if ( Gpsdata == '*' && com_cnt >= 6) {
        com_cnt = 0;                        // end of GPRMC message
        lat_cnt = 0;
        log_cnt = 0;
```

```
        lat_cntP = 0;
        log_cntP = 0;
        AorVdata = 0;
        flg      = 0;
        finish   = 1;

      }
    }
  }
  Serial.print("Latitude  : ");
  Serial.println(lat);
  Serial.print("Longitude : ");
  Serial.println(lg);


  float Latnum = atof(lat);
  float Longnum = atof(lg);

  degWholeLat = float(int(Latnum / 100)); //gives me the whole degree part
of Longitude
  degDecLat = (Latnum - degWholeLat * 100) / 60; //give me fractional part
of longitude
  degLat = degWholeLat + degDecLat; //Gives complete correct decimal form
of Longitude degrees
  if (latP == 'W') { //If you are in Western Hemisphere, longitude degrees
should be negative
    degLat = (-1) * degLat;
  }

  degWholeLong = float(int(Longnum / 100)); //gives me the whole degree
part of Longitude
  degDecLong = (Longnum - degWholeLong * 100) / 60; //give me fractional
part of longitude
  degLong = degWholeLong + degDecLong; //Gives complete correct decimal
form of Longitude degrees
  if (lgP == 'S') { //If you are in Western Hemisphere, longitude degrees
should be negative
    degLong = (-1) * degLong;
  }

  Serial.print("Latitude Position: ");
  Serial.println(latP);
  Serial.print("Longitude Position : ");
  Serial.println(lgP);
  Serial.print("STATUS : ");
  Serial.println(AorV);
  Serial.print("Latitude Position in Dec: ");
  Serial.println(degLat);
  Serial.print("Longitude Position in Dec : ");
  Serial.println(degLong);

  while (sendflg == 0) {
    if (smsflag == 0 && finish == 1)         /* Send message after storing
GPS cordinates               */
    {
      Serial.print("AT+CMGS=\"+91xxxxxxxxxx \"\r\n");
      //Serial.print("AT+CMGS=\"+91xxxxxxxxxx\"\r\n");   /* Replace
xxxxxxxxxx with a valid 10-digit mobile no:     */
      delay(1000);
```

```
    }

    while (Serial.available())                          /* Check if any data
has arrived in hardware UART          */
    {
      Rec_data = Serial.read();                         /* Copy the received
character to a variable               */
      if (Rec_data == '>')                              /* Set flag for
response to "AT+CMGS=\"+91xxxxxxxxxx\""    */
      {
        smsflag = 1;
      }
    }

    // Serial.println("Inside SMS");
    if (smsflag == 1)                       /* Send message after storing GPS
cordinates               */
    {
      //Serial.print("AT+CMGS=\"+91xxxxxxxxxx \"\r");    /* Replace
xxxxxxxxxx with a valid 10-digit mobile no:     */
      Serial.println("Vehicle successfully located at :");
      Serial.print("http://www.google.com/maps/search/");
      Serial.print(degLat, 4);                          /* Send
latitude                                          */
      Serial.print(",");
      // Serial.print("Longitude:");
      Serial.print(degLong, 4);                         /* Send
longitude                                         */
      Serial.print("/@");
      Serial.print(degLat, 4);                          /* Send
latitude                                        */
      Serial.print(",");
      // Serial.print("Longitude:");
      Serial.print(degLong, 4);
      Serial.print(",15z");
      Serial.print('\x1A');                             /* Send Ctrl+Z after
the message                           */
      sendflg = 1;
    }
  }
}
```

Applications:

- Vehicle tracking system

Conclusion:

A stand-alone vehicle tracking system is developed with location access only to authorized sender's mobile.

References:

[1] http://www.circuitstoday.com/interface-gsm-module-with-arduino

[2]http://www.rhydolabz.com/documents/SIM800%20Series_AT%20Command%20Manual_V1.09.pdf

[3] http://www.rhydolabz.com/wiki/?p=10551

Experiment 19: Bluetooth AT Commands

Aim:

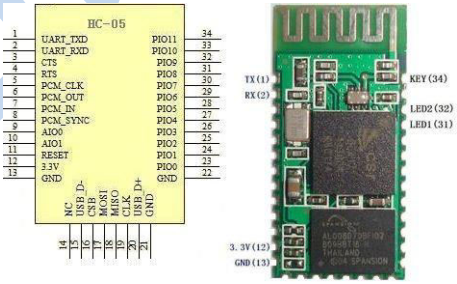
To check various AT commands available for HC 05 Bluetooth Module.

Objective:

To explore and learn about the AT commands provided for the given module.

Components Required:

1. HC 05 Bluetooth Module
2. Arduino UNO board
3. Connecting wires



Procedure:

1. Connections and entering AT mode

- Connect RX pin of the Bluetooth module to pin 11 of Arduino UNO.
- Connect TX pin of Bluetooth to pin 10 of Arduino.
- Connect KEY pin of module to pin 9 of Arduino UNO and GND to GND.
- Solder a wire to the 34th pin of the HC 05 module and connect it to pin 9 of Arduino or you can just hold a wire (the pointy end of a male to male jumper wire would be convenient) while connecting the VCC to 5V.
- Now the LED on the BT module will blink at the interval of 2 seconds. This means it has entered the AT mode. If the LED blinks faster, then AT mode was not entered. Disconnect VCC, check your circuit and try again.
- Once the BT module is in AT mode, you can release the wire.

2. Upload the given code on to the Arduino

Upload the code to the Arduino board, once the code is uploaded, open Serial monitor, change "no line ending" to "both NL & CR" and baud to 38400 at the bottom, close it and disconnect Arduino board. Now again reconnect Arduino, connect VCC of Bluetooth module to Arduino 5V, and open serial monitor. The led on the module should blink at the interval of 2 seconds. That means it has entered the AT mode. Now you are ready to enter AT commands.

3. AT Commands

Type "AT" (without the quotes) on the serial monitor and press enter. If "OK" appears then everything is all right and the module is ready to take command. Now you can change the name of the module, retrieve address or version or even reset to factory settings.

Most useful AT commands are:

AT : Check the connection.

AT+NAME : See default name

AT+ADDR : See default address

AT+VERSION : See version

AT+UART : See baud rate

AT+ROLE: See role of BT module(1=master/0=slave)

AT+RESET : Reset and exit AT mode

AT+ORGL : Restore factory settings

AT+PSWD: See default password

Source Code:

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // RX, TX
void setup() {
Serial.begin(9600);
pinMode(9,OUTPUT); digitalWrite(9,HIGH);
Serial.println("Enter AT commands:");
mySerial.begin(38400);
}
void loop()
{
if (mySerial.available())
Serial.write(mySerial.read());
if (Serial.available())
mySerial.write(Serial.read());
}
```

Applications:

- Various functions like resetting BT module name, baud, seeing all paired devices, power modes etc. can be carried out using AT commands.

Conclusion:

The HC 05 Bluetooth module can be custom configured using the given steps using AT commands.

References:

[1] http://www.instructables.com/id/AT-command-mode-of-HC-05-Bluetooth-module/?ALLSTEPS

Experiment 20: LED Control via Bluetooth using Arduino

Aim:

To control an LED using an Arduino UNO, an Android app and Bluetooth.

Objective:

To be able to create useful of Internet of things based projects.

Components Required:

1. Arduino UNO
2. LED
3. Resistor 221ohm
4. Android phone
5. HC 05 Bluetooth module

Apps and Online Services:
1. LED controller app
2. Arduino IDE

Connections:

Arduino Pins	HC 05 Pins
RX (Pin 0)	TX
TX (Pin 1)	RX
5V	VCC
GND	GND

Connect a LED negative to GND of Arduino and positive to pin 13 with a resistance valued between 220Ω – 1KΩ.

Procedure:

HC 05/06 works on serial communication. The android app is designed sending serial data to the Bluetooth module when certain button is pressed. The Bluetooth module at another end receive the data and send to Arduino through the TX pin of Bluetooth module (RX pin of Arduino). The
Code fed to Arduino check the received data and compares. If received data is 1 the LED turns on turns OFF when received data is 0
Open the serial monitor and watch the received data

How to use the App?

1. Download the Application from reference [1].
2. Pair your device with HC 05/06 Bluetooth module
1) Turn ON HC 05/06 Bluetooth module
2)Scan for available device
3) Pair to HC 05/06 by entering default password 1234 OR 0000
3. Install LED application on your android device
4. Open the Application
5. Press paired devices
6. Select your Bluetooth module from the List (HC 05)
7. After connecting successfully
8. Press ON button to turn ON LED and OFF button to turn OFF LED
9. Disconnect button to disconnect from Bluetooth module

Source Code:

```
/*
* Bluetooth Basic: LED ON OFF
* Download the App : https://github.com/Mayoogh/Arduino-Bluetooth-Basic
* This program lets you to control a LED on pin 13 of Arduino using a bluetooth module
*/
char data = 0; //Variable for storing received data
void setup()
{
Serial.begin(9600); //Sets the baud for serial data transmission
pinMode(13, OUTPUT); //Sets digital pin 13 as output pin
}
void loop()
{
if(Serial.available() > 0) // Send data only when you receive data:
{
data = Serial.read(); //Read the incoming data & store into data
Serial.print(data); //Print Value inside data in Serial monitor
Serial.print("\n");
if(data == '1') // Checks whether value of data is equal to 1
digitalWrite(13, HIGH); //If value is 1 then LED turns ON
else if(data == '0') // Checks whether value of data is equal to 0
digitalWrite(13, LOW); //If value is 0 then LED turns OFF
}
}
```

Applications:

- Remote access to street lights
- Remote access to an industry/factory lights
- Home Automation

Conclusion:

With the use of an app and Bluetooth, we can conclude that the above mentioned applications can be fulfilled efficiently.

References:

[1] https://github.com/Mayoogh/Arduino-Bluetooth-Basic
[2] http://www.robotshop.com/media/files/pdf/rb-ite-12-bluetooth_hc05.pdf
[3] https://create.arduino.cc/projecthub/user206876468/arduino-bluetooth-basic-tutorial-d8b737

Experiment 21: Control LED brightness via Bluetooth and Arduino

Aim:

To control LED brightness using an Android app via Bluetooth.

Objective:

To learn wireless communication(Bluetooth) between devices.

Components Required:

1. Arduino UNO
2. LED
3. Resistor 221ohm
4. Android phone
5. HC 05 Bluetooth module

Apps and Online Services:
1. RoboRemo Android app
2. Arduino IDE

Connections:

Arduino Pins	HC 05 Pins
RX (Pin 0)	TX
TX (Pin 1)	RX
5V	VCC
GND	GND

Connect the LED negative to GND of Arduino and positive to pin 9 with a resistance valued between 220Ω – 1KΩ.

Procedure:

HC 05/06 works on serial communication. The android app is designed to send serial data to the Bluetooth module when the slider is released. The Bluetooth module at the other end receives the data and sends to Arduino through the TX pin of Bluetooth module (RX pin of Arduino). The Code fed to Arduino checks the received data and assigns that value as the LED intensity value at output pin 9.
The incoming data can be viewed on the serial monitor.

How to use the App?

1. Pair your device with HC 05/06 Bluetooth module.
   i. Turn ON HC 05/06 Bluetooth module.
   ii. Scan for available device.
   iii. Pair to HC 05/06 by entering default password 1234 OR 0000.
2. Install LED application on your android device.
3. Open the Application.
4. Press paired devices.
5. Select your Bluetooth module from the List (HC 05) and connect.
6. From the menu, add a slider to the view and give it a label 'L'.
7. Slide the knob along the length to send different intensity values.
8. Observe LED brightness vary accordingly.
9. Disconnect button to disconnect from Bluetooth module.

Source Code:

```
/* Example to change brightness of LED
using RoboRemo apk on
android via Bluetooth Communication
Edited by:Interns */

String fadeAmount;    // example string
int brightness;    // how bright the LED is
int led = 9;          // the pin that the LED is attached to. It is a PWM enabled pin

void setup() {
  Serial.begin(9600);
  pinMode(led, OUTPUT);
  digitalWrite(led, LOW);
}

void loop() {
  if (Serial.available() > 0) {
    fadeAmount = Serial.readString();
    //Serial.println(fadeAmount);// Print the initial string
    fadeAmount.remove(0, 2); // Remove two characters starting at index=0
    //Serial.println(fadeAmount); // Print the cut string
    brightness = fadeAmount.toInt();
    Serial.println(brightness);
    analogWrite(led, brightness);
  }
}
```

Applications:

- The intensity of tube lights etc. can be varied using just an app.

- The speed of motor related appliances like fans, robots, turbines can also be controlled using an app.

Conclusion:

With the use of an app and Bluetooth, we can conclude that the above mentioned applications can be fulfilled efficiently.

References:

[1] https://www.arduino.cc/

Experiment 22: Voice Commands via Bluetooth

Aim:

To control LED by giving voice commands for ON/OFF and intensity by Android app via Bluetooth communication.

Objective:

To learn wireless communication(Bluetooth) using voice commands between devices.

Components Required:

1. Arduino UNO
2. LED
3. Resistor 221ohm
4. Android phone
5. HC 05 Bluetooth module

Apps and Online Services:
1. Arduino Bluecontrol Android app
2. Arduino IDE

Connections:

Arduino Pins	HC 05 Pins
RX (Pin 0)	TX
TX (Pin 1)	RX
5V	VCC
GND	GND

Connect the LED negative to GND of Arduino and positive to pin 13 with a resistance valued between 220Ω – 1KΩ.

Procedure:

HC 05/06 works on serial communication. The android app is designed to send serial data to the Bluetooth module when the voice command is received. The Bluetooth module at the other end receives the data and sends to Arduino through the TX pin of Bluetooth module (RX pin of Arduino). The
Code fed to Arduino checks the received data and turns LED ON/OFF at output pin 13.
The incoming data can be viewed on the serial monitor.

How to use the App?
1. Pair your device with HC 05/06 Bluetooth module.

iv. Turn ON HC 05/06 Bluetooth module.
v. Scan for available device.
vi. Pair to HC 05/06 by entering default password 1234 OR 0000.

2. Install the application on your android device.
3. Open the Application.
4. Press paired devices.
5. Select your Bluetooth module from the List (HC 05) and connect.
6. Open the 'Voice Control' tab from the options.
7. In the menu, add the data to be associated to the respective voice command e.g. voice command ON can be given data 1 and command OFF can be given data 0.
8. When one of these commands are recognized, the corresponding data is sent to the Arduino via the HC 05.
9. Comparing the incoming data, the LED can be observed to turn ON/OFF.
10. This could also be done to control LED intensity by assigning corresponding values '1' and '0' to voice commands 'up' and 'down' and connecting LED to PWM pin 9.

Source Code:

Code for ON/OFF

```
char incomming;
void setup() {
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}

void loop() {
  if (Serial.available() > 0) {
    incomming = Serial.read();
    Serial.println(incomming);// Print the initial string

    if (incomming == '1') {
      Serial.println("lights on");
      digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
      delay(1000);              // wait for a second
    }
    else if (incomming == '0') {
      Serial.println("lights off");
      digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
      delay(1000);
    }
    delay(500);
  }
}
```

Code for LED intensity

```
char incomming;
```

```
int brightness = 128;
void setup() {
  Serial.begin(9600);
  pinMode(9, OUTPUT);
}

void loop() {
  if (Serial.available() > 0) {
    incomming = Serial.read();
    Serial.println(incomming);// Print the initial string

    if (incomming == '1') {
      Serial.println("increasing intensity");
      brightness = brightness + 5;
      analogWrite(9, brightness);
      delay(1000);            // wait for a second
    }
    else if (incomming == '0') {
      Serial.println("decreasing intensity");
      brightness = brightness - 5;
      analogWrite(9, brightness);
      delay(1000);
    }
    delay(500);
  }
}
```

Applications:

- Any appliances can be turned ON/OFF using only voice commands.
- This can greatly help physically disabled people carry out simple jobs.

Conclusion:

With the use of an app and Bluetooth, we can conclude that the above mentioned applications can be fulfilled efficiently.

References:

[1] https://www.arduino.cc/

Experiment 23: Cross Protocol Communication

Aim:

To transfer data over between 2 protocols (Bluetooth and WiFi).

Objective:

To learn cross protocol communication.

Components Required:

1. Arduino Yun
2. HC 05 Bluetooth Module
3. ESP 8266 WiFi Module
4. Breadboard & Jumper Wires

Apps and Online Services:
1. Arduino Bluetooth app
2. Arduino IDE

Connections:

Connections of HC 05:

Arduino Pins	HC 05 Pins
RX (Pin 0)	TX
TX (Pin 1)	RX
5V	VCC
GND	GND

Connections of ESP8266:

Arduino Pins	ESP8266 Pins
GND	GND
D3	RX
3v3	VCC
3v3	CH_PD
D2	TX

Procedure:

1. Create a channel on Thingspeak using your account and note down the WriteAPI key.
2. Download the Android app on your phone.
3. Make the circuit connections as given above and pair your Android phone with the HC 05 Bluetooth module.

4. Upload the source code to the Arduino UNO and observe the incoming data on the serial monitor.
5. Send characters from the app to the Arduino via Bluetooth and observe your Thingspeak channel.
6. Refresh the page and you will observe the ASCII value of the received characters plotted on a time varying graph.

Source Code:

```
#include <SoftwareSerial.h>

SoftwareSerial serb(10, 11);  // software serial #1: RX = digital pin 10, TX = digital pin 11
SoftwareSerial serw(2, 3);  // software serial #2: RX = digital pin 8, TX = digital pin 9

int strVolt;
String apiKey = "F3MDLY1MJVZG2EWT"; // replace with your channel's thingspeak API key

void setup() {
 Serial.begin(9600); // Open serial communications and wait for port to open
 while (!Serial) {
  ; // wait for serial port to connect. Needed for native USB port only
 }
 // Start each software serial port
 serw.begin(9600);
 serb.begin(9600);
}

void loop() {
 // By default, the last intialized port is listening.
 // when you want to listen on a port, explicitly select it:
 serb.listen();
 Serial.println("Waiting for Bluetooth input");
 // while there is data coming in, read it and send to the hardware serial port:
 while (serb.available() > 0) {
  int strVolt = serb.read();
  Serial.println("Bluetooth received :");
  Serial.write(strVolt);
  // blank line to separate data from the two ports:
  Serial.println();
 // Now listen on the second port
  serw.listen();
  Serial.println("Uploading this data :");
  Serial.print(strVolt);
  Serial.println(" C");
```

```
  // TCP connection
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "184.106.153.149"; // api.thingspeak.com
  cmd += "\",80";
  serw.println(cmd);
  Serial.println(cmd);

  if (serw.find("ERROR")) {
   Serial.println("AT+CIPSTART error");
   return;
  }
 // prepare GET string
  String getStr = "GET /update?api_key=";
  getStr += apiKey;
  getStr += "&field1=";
  getStr += String(strVolt);
  getStr += "\r\n\r\n";
  // send data length to initiate connection
  cmd = "AT+CIPSEND=";
  cmd += String(getStr.length());
  serw.println(cmd);
  Serial.println(cmd);
  //The > sign indicates that connection is successful and insert data to be sent
  if (serw.find(">")) {
   serw.print(getStr);
   Serial.println(getStr);
  }
  else if (serw.find("ERROR")) {
   Serial.println(" error");
  }
  else {
   // alert user
   serw.println("AT+CIPCLOSE");
   Serial.println("AT+CIPCLOSE");
  }
  serw.flush();
  delay(16000);// thingspeak needs 15 sec delay between updates
  Serial.println(); // blank line to separate data from the two ports:
 }
}
```

Applications:

- Any action in a certain home automation system can be remotely seen on the internet.
- This can be used for home security solutions, to keep a tab of which appliances were used etc. in the absence of the owner.

Conclusion:

Using Thingspeak, a workable prototype of an IoT project was created. It can be concluded that Arduino, along with other sensors and modules, can create many successful IoT projects.

References:

[1] http://community.thingspeak.com/tutorials/arduino/send-data-to-thingspeak-with-arduino/
[2] https://www.youtube.com/watch?v=sPfJ9f1FuUE

Experiment 24: Arduino and Modbus Communication

Aim:

To view sensor data obtained from Arduino using Modbus communication protocol.

Objective:

To learn about Modbus communication protocol and send data from Arduino to Modbus application program on computer.

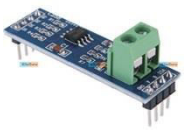
Components Required:

1. Arduino UNO
2. RS485 to USB
3.TTL to RS485 Module. (Max485)
4. Jumper Wires
5. IR obstacle sensor

Connections:

Connections with Arduino and MAX485 module:

Arduino Pins	MAX485 module
Rx0	R0
Tx1	DI
D2	DE+RE (shorted)
5v	VCC
GND	GND



Connections with MAX485 module and RS485 to USB:

MAX485 module	RS485 to USB
A	Tx
B	Rx

Connections with Arduino and IR obstacle detection module:

Arduino Pins	IR obstacle module
A0	O/P
5v	VCC
GND	GND

1. Perform the following connections as shown in the table.

About Modbus:

Modbus is a serial communication protocol developed by Modicon published by Modicon® in 1979 for use with its programmable logic controllers (PLCs). In simple terms, it is a method used for transmitting information over serial lines between electronic devices. The device requesting the information is called the Modbus Master and the devices supplying information are Modbus Slaves. In a standard Modbus network, there is one Master and up to 247 Slaves, each with a unique Slave Address from 1 to 247. The Master can also write information to the Slaves.

Procedure:

1. Once you have your Arduino IDE installed and updated to the latest version, we need to download an external library which allows Arduino to act as a Modbus slave.
2. Go to link provided in Reference [1] and download the e Modbus Slave library.
3. Next in the Arduino IDE, click on :
4. *Sketch >> Include Library >> Add .ZIP library.*
5. Browse and select the folder where you downloaded the library.
6. Now plug in your Arduino without the connections made. Create a new sketch and copy paste the code contents provided below.
7. Your Arduino board is ready with the program burnt in it. Next, in order to see the output, we need to download and install a Modbus application program on our Computer.
8. Go to link provided in Reference [5] and download and install the QModbus application.
9. Perform all the connections with the Arduino.
10. Open the QModbus application and follow along.
11. In the QModbus application, change the Function Code to *Read holding Registers.*
12. Click on *Options >> Modbus RTU.* Change to the following settings:
    Serial port : *COMxx*(According to yours)
    Baud : *9600*
    Data Bits : *8*
    Parity : *None*
13. After performing the required settings, click on *Connect.*
14. Output to be observed is 1 or 0, based on the obstacle detection of IR module.

Source Code:

```
/**
 *Modbus and Arduino Communication
 *Code modified by : IoT Intern, 2016
 *http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/arduino-ir-obstacle-sensor-tutorial-and-manual/
 *https://www.youtube.com/watch?v=5NHtQTJGrmo
**/
```

```
#include <SimpleModbusSlave.h>
#define  LED 9
int LEDir = 13; // Use the onboard Uno LED
int isObstaclePin = 7;  // This is our input pin
int isObstacle = HIGH;  // HIGH MEANS NO OBSTACLE

/////////////// registers of your slave ///////////////////
enum
{
 // just add or remove registers and your good to go...
 // The first register starts at address 0
 ADC_VAL,
 PWM_VAL,
 HOLDING_REGS_SIZE // leave this one
 // total number of registers for function 3 and 16 share the same register array
 // i.e. the same address space
};

unsigned int holdingRegs[HOLDING_REGS_SIZE]; // function 3 and 16 register array

void setup() {
 pinMode(LEDir, OUTPUT);
 pinMode(isObstaclePin, INPUT);
 Serial.begin(9600);
 modbus_configure(&Serial, 9600, SERIAL_8N2, 1, 2, HOLDING_REGS_SIZE, holdingRegs);

 // modbus_update_comms(baud, byteFormat, id) is not needed but allows for easy update of the
 // port variables and slave id dynamically in any function.
 modbus_update_comms(9600, SERIAL_8N2, 1);
 pinMode(LED, OUTPUT);
}


void loop() {


 isObstacle = digitalRead(isObstaclePin);
 modbus_update();

 holdingRegs[ADC_VAL] = analogRead(A0); // update data to be read by the master to adjust the PWM

 analogWrite(LED, holdingRegs[PWM_VAL] >> 2); // constrain adc value from the arduino master to 255

 if (isObstacle == HIGH)
 {
  Serial.println("OBSTACLE!!, OBSTACLE!!");
  digitalWrite(LED, HIGH);
 }
 else
 {
  Serial.println("clear");
  digitalWrite(LED, LOW);
 }
 delay(1000);
}
```

Applications:

- Multiple sensor data values can be easily obtained and a vast data acquisition system can be implemented using Modbus protocol.

Conclusion:

Modbus protocol used for serial communication is used to obtain and view sensor data from Arduino.

References:

[1] https://github.com/angeloc/simplemodbusng

[2] http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/arduino-ir-obstacle-sensor-tutorial-and-manual/

[3] https://www.youtube.com/watch?v=5NHtQTJGrmo

[4] http://www.simplymodbus.ca/FAQ.htm#Modbus

[5] https://sourceforge.net/projects/qmodmaster/

Experiment 25: Upload data on Thingspeak using Arduino and ESP8266

Aim:

To send sensor data to Thingspeak using Arduino UNO and ESP8266.

Objective:

To learn to upload data to the internet via the WiFi module.

Components Required:

1. Arduino UNO
2. ESP8266 WiFi Module
3. LM35 Temperature Sensor
4. Connecting wires

Connections:

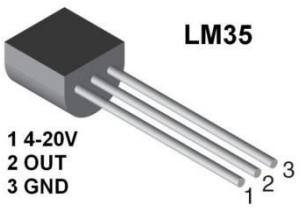
Connections of the ESP8266:

Arduino Pins	ESP8266 Pins
GND	GND
D3	RX
3v3	VCC
3v3	CH_PD
D2	TX

Pins D2 & D3 are the assigned software serial port.

Connections of the LM35:

Arduino Pins	LM35 Pins
5V	VCC
A0	OUT
GND	GND



Procedure:

I. Create a channel on Thingspeak

ThingSpeak requires a user account and a channel. A channel is where you send data and where ThingSpeak stores data. Each channel has up to 8 data fields, location fields, and a status field. You

can send data every 15 seconds to ThingSpeak, but most applications work well every minute.

- Sign up for new *User Account* – https://www.thingspeak.com/users/new
- Create a new *Channel* by selecting *Channels*, *My Channels*, and then *New Channel*
- Note the *Write API Key* and *Channel ID*

II. Copy the code in the sketch

Copy the code in the Arduino IDE sketch and replace the 'apiKey=' with your *Write API Key* obtained in the previous step.

III. Make the connections

Make the connections as given above and upload the code into the Arduino. Open your Thingspeak channel and observe the data in graphical representation under the Private View tab.

Source Code:

```
#include <SoftwareSerial.h>

// LED
int ledPin = 13;
int sensor_pin=A0;  // variable for sensor
float sample=0;
float temp =0;     // for temperature

// replace with your channel's thingspeak API key
String apiKey = "SJE3XM4D90Y794LG";

// connect 2 to TX of Serial USB
// connect 3 to RX of serial USB
SoftwareSerial ser(2,3); // RX, TX

// this runs once
void setup() {
  // initialize the digital pin as an output.
  pinMode(ledPin, OUTPUT);
  // enable debug serial
  Serial.begin(115200);
  // enable software serial
  ser.begin(115200);
  // reset ESP8266
  ser.println("AT+RST");
}

// the loop
void loop() {
  // blink LED on board
```

```
  digitalWrite(ledPin, HIGH);
  delay(200);
  digitalWrite(ledPin, LOW);
  voltage();
  esp_8266();
}

void voltage()
{
  sample = analogRead(sensor_pin);
  temp = (5.0*sample*100.0)/1024;
}

void esp_8266()
{
 // convert to string
  char buf[32];
  String strVolt = dtostrf( temp, 4, 1, buf);
 // String strVolt = "5";
  Serial.print(strVolt);
  Serial.println(" C");
  // TCP connection
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "184.106.153.149"; // api.thingspeak.com
  cmd += "\",80";
  ser.println(cmd);
  if(ser.find("Error")){
    Serial.println("AT+CIPSTART error");
    return;
  }
  // prepare GET string
  String getStr = "GET /update?api_key=";
  getStr += apiKey;
  getStr +="&field1=";
  getStr += String(strVolt);
  getStr += "\r\n\r\n";
  // send data length
  cmd = "AT+CIPSEND=";
  cmd += String(getStr.length());
  ser.println(cmd);
  if(ser.find(">")){
    ser.print(getStr);
  }
  else{
```

```
   ser.println("AT+CIPCLOSE");
   // alert user
   Serial.println("AT+CIPCLOSE");
  }
  // thingspeak needs 15 sec delay between updates
  delay(16000);
}
```

Applications:

- Weather station for an area could be created and kept a tab on.
- Any other data can be remotely viewed and accessed using this project.

Conclusion:

Using Thingspeak, a workable prototype of an IoT project was created. It can be concluded that Arduino, along with other sensors, can create many successful IoT projects.

References:

[1] http://community.thingspeak.com/tutorials/arduino/send-data-to-thingspeak-with-arduino/
[2] https://www.youtube.com/watch?v=sPfJ9f1FuUE

Experiment 26: Bluetooth controlled Wireless Bot using Arduino

Aim:

To develop a wireless bot using Arduino and Bluetooth.

Objective:

To learn to use Bluetooth with Arduino and control a bot wirelessly from the smart phone.

Components Required:

1. Arduino UNO
2. HC05 bluetooth module
3. Motor Driver module L293d
4. Soldering iron and solder wire
5. Connecting Wires
6. 9v dc battery
7. Chassis
8. DC Motor 100 rpm
9. DC wheels

Connections:

Connections with the HC05 Bluetooth Module :

Arduino Pins	HC05 Bluetooth
5v	**5v**
GND	GND
Rx	Tx
Tx	Rx

Connections with the Motor Driver module:

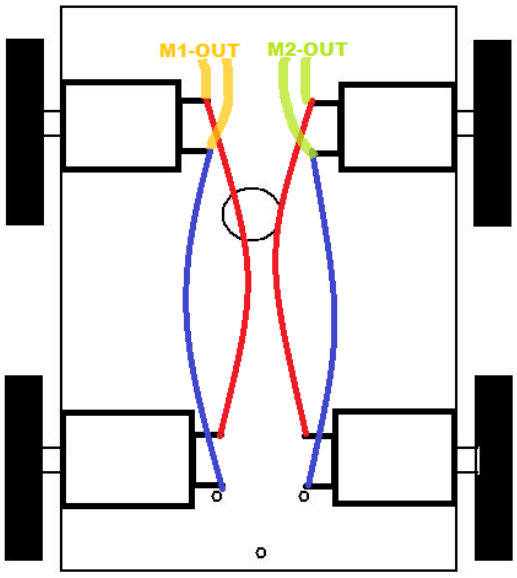
Arduino Pins	Motor Driver L293d
5	M1-IN
6	M1-IN
10	M2-IN
11	M2-IN

3. Perform the following connections as shown in the table.
4. Connect one DC motor to the M1-OUT and the other DC Motor to M2-OUT.
5. Also apply 9V DC voltage to the motor driver module.
6. In order to power the Arduino, connect 9v battery to a DC jack as shown in the figure below:

Note: Make sure the GND of motor driver module is also connected to the GND of Arduino.

Procedure:

- Let's first keep the software ready before starting with the Hardware. Copy paste the code given below and upload it to the Arduino.
- Now time to get our Hardware ready. The connections of the DC gear motor are clearly shown in the diagram below. Solder the terminals correctly and then connect it to the Motor Driver module as shown.



- Now once your bot is ready, connect the HC05 Module to the Arduino as given in the table above.
- Connect the M1-In and M2-IN of the Motor Driver module to the Arduino pins as given in the Table above. Make sure you interconnect all the grounds.
- Fit all the components including the batteries upon the chassis.

- Download the app whose link is provided in reference [1] *BlueTooth Electronics*.
- Open the App and follow the steps. *New panel >> Edit >> Buttons >> Drag and drop a total of 9 buttons >> Select the button you want to go full speed forward >> Edit >> (In the pop up window) Press text : 0*
- *Leave the Release Text field empty for all the buttons.*
- *Similarly do it for all.*
    - *full speed backward >> Edit >> Press text : 3*
    - *full speed right >> Edit >> Press text : 1*
    - *full speed left >> Edit >> Press text : 2*
    - *slow speed forward >> Edit >> Press text : F*
    - *slow speed backward >> Edit >> Press text : B*
    - *slow speed left >> Edit >> Press text : L*
    - *slow speed right >> Edit >> Press text : R*
    - *stop >> Edit >> Press text : W*
- Then, *Connect >> Discover >> Pair >> Connect >> Done >> Select your panel >> Run!*
- *Debugging: Measure the voltage with the help of a multimeter across the Motor Driver module.*
- Your wireless Bluetooth controlled bot is ready to speed.

Source Code:

```
/**
   Bluetooth Controlled Wireless Bot
   Code developed by : Ms. Michelle D'Souza, IoT Intern, 2016.
   Grateful to the whole open source community on internet.
*/

char inComming;
#define m11 5
#define m12 6
#define m21 10
#define m22 11

void setup() {
  Serial.begin(9600);
```

```
  pinMode(m11, OUTPUT);
  pinMode(m12, OUTPUT);
  pinMode(m21, OUTPUT);
  pinMode(m22, OUTPUT);
  pinMode(13, OUTPUT);

}

void loop() {
  if (Serial.available() > 0) {
    inComming = Serial.read();
    Serial.println(inComming);

    if (inComming == '0') {
      front();
    }

    else if (inComming == '3') {
      back();
    }

    else if (inComming == '2') {
      left();
    }
    else if (inComming == '1') {
      right();
    }
    else  if (inComming == 'F') {
      slowfront();
    }

    else if (inComming == 'B') {
      slowback();
    }

    else if (inComming == 'L') {
      slowleft();
    }
    else if (inComming == 'R') {
      slowright();
    }

    else {
      dontMove();

      Serial.println("not moving");

    }
  }

}

void front() {
  digitalWrite(m12, HIGH);
  digitalWrite(m11, LOW);

  digitalWrite(m22, HIGH);
  digitalWrite(m21, LOW);
  Serial.println("front");
```

```
  //delay(1000);


}
void back() {
  digitalWrite(m11, HIGH);
  digitalWrite(m12, LOW);
  digitalWrite(m21, HIGH);
  digitalWrite(m22, LOW);
  Serial.println("back");

  //delay(1000);

}

void left() {
  digitalWrite(m11, LOW);
  digitalWrite(m12, LOW);

  digitalWrite(m22, HIGH);
  digitalWrite(m21, LOW);
  Serial.println("left");

  //delay(1000);

}

void right() {
  digitalWrite(m12, HIGH);
  digitalWrite(m11, LOW);
  digitalWrite(m22, LOW);
  digitalWrite(m21, LOW);
  Serial.println("right");

  //delay(1000);

}

void dontMove() {
  digitalWrite(m11, LOW);
  digitalWrite(m12, LOW);
  digitalWrite(m22, LOW);
  digitalWrite(m21, LOW);
  Serial.println("stop");

  //delay(1000);

}
void slowfront() {
  analogWrite(m12, 150);
  digitalWrite(m11, LOW);

  analogWrite(m22, 150);
  digitalWrite(m21, LOW);
  Serial.println("slow front");

  //delay(1000);
```

```
}
void slowback() {
  analogWrite(m11, 150);
  digitalWrite(m12, LOW);
  analogWrite(m21, 150);
  digitalWrite(m22, LOW);
  Serial.println("slow back");

  //delay(1000);

}

void slowleft() {

  digitalWrite(m11, LOW);
  digitalWrite(m12, LOW);

  analogWrite(m22, 150);
  digitalWrite(m21, LOW);
  Serial.println("slow left");

  //delay(1000);

}

void slowright() {
  analogWrite(m12, 150);
  digitalWrite(m11, LOW);
  digitalWrite(m22, LOW);
  digitalWrite(m21, LOW);
  Serial.println("slowright");
  //delay(1000);

}
```


Applications:

- Security and monitoring system with camera provisioning.

Conclusion:

With the use of Bluetooth and Arduino, wireless controlled bot is developed.

References:

[1] https://play.google.com/store/apps/details?id=com.keuwl.arduinobluetooth&hl=en

[2] http://www.instructables.com/id/Arduino-AND-Bluetooth-HC-05-Connecting-easily/

Experiment 27: Log sensor data from Arduino to Excel sheet

Aim:

To log sensor data from Arduino to excel sheet of computer.

Objective:

To learn how to collect huge amounts of sensor data for further processing.

Components Required:

7. Arduino UNO
8. HC – SR04 ultrasonic sensor
9. Connecting wires

About the sensor:

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package and can detect from 2cm to 400 cm or 1" to 13 feet. It operation is not affected by sunlight or black material. It comes complete with ultrasonic transmitter and receiver module.

Features

Power Supply: +5V DC

Quiescent Current: <2mA

Working Current: 15mA

Effectual Angle: <15°

Ranging Distance: 2cm – 400 cm/1" – 13ft

Resolution: 0.3 cm

Measuring Angle: 30 degree

Trigger Input Pulse width: 10uS

Dimension: 45mm x 20mm x 15mm

Pins

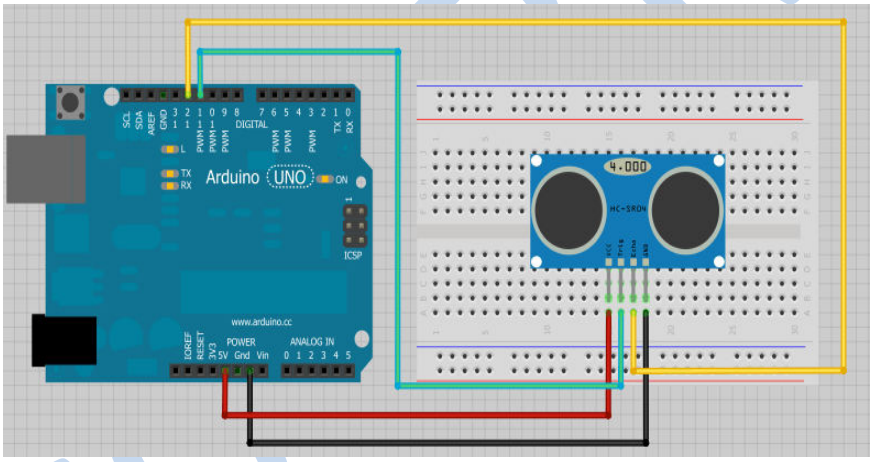
VCC: +5VDC

Trig: Trigger (INPUT)

Echo: Echo (OUTPUT)

GND: GND

Connections:



Connections with Arduino UNO

Procedure:

10. In order to log data to excel sheet from Arduino, we need to first download and install an external software.
11. Please go to reference 1 and download and install the software on your computer according to the specifications of your system. (Windows, Linux, Mac, etc.)
12. Now open the Application just installed *PLX-DAQ* and select the COM port number and set Baud rate to 9600.
13. Next connect the Ultrasonic sensor as shown in the image above and upload the source code given below.

Source Code:

```
/**
  Log Data to Excel Sheet
  Code by : IoT Intern, 2016
  http://www.instructables.com/id/Sending-data-from-Arduino-to-Excel-and-plotting-it/?ALLSTEPS
**/

#define trigPin 7
#define echoPin 6

void setup() {
 //always starts in line 0 and writes the thing written next to LABEL

 pinMode(13, OUTPUT);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);

 Serial.begin(9600); // the bigger number the better
 Serial.println("CLEARDATA"); //clears up any data left from previous projects
 Serial.println("LABEL,Time,Timer,Distance,Sensor Value"); //always write LABEL, so excel knows the next
things will be the names of the columns
 Serial.println("RESETTIMER"); //resets timer to 0

}

void loop() {

 int just4Demo = 5;
 int duration, distance;
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
 duration = pulseIn(echoPin, HIGH);
 distance = (duration / 2) / 29.1;

 Serial.print("DATA,TIME,TIMER,"); //writes the time in the first column A and the time since the
measurements started in column B
 Serial.print(distance);
 Serial.print(",");
 Serial.print(just4Demo);
 Serial.println(); //be sure to add println to the last command so it knows to go into the next row on the
second run
 delay(1000); //add a delay

}
```

Applications:

- Data acquisition system: The data obtained from any sensor is important in case of IoT applications. This data obtained can be continuously stored in excel sheet of computer for future reference.

Conclusion:

Data from any sensor can be logged to excel sheet of computer with ease.

References:

[4] https://www.parallax.com/downloads/plx-daq
[5] http://www.micropik.com/PDF/HCSR04.pdf
[3] http://www.instructables.com/id/Sending-data-from-Arduino-to-Excel-and-plotting-it/?ALLSTEPS

Experiment 28: MATLAB Visualization using Thingspeak

Aim:

To get MATLAB visualisations on Thingspeak channel data.

Objective:

To integrate MATLAB with Thingspeak.

Components Required:

1. Arduino Yun board
2. Micro-USB cable

Procedure:

1. Open your Thingspeak account and open a channel with data.
2. In your channel, select the tab MATLAB Visualisation.
3. Select View Temperature Variation via Histogram and hit Create.
4. A MATLAB code will appear, select Save and Run.
5. A histogram will be obtained.

Source Code:

```
% Read temperature for the last 10 hours from a ThingSpeak channel and
% visualize temperature variations using the MATLAB HISTOGRAM function.

% Channel 12397 contains data from the MathWorks Weather Station, located
% in Natick, Massachusetts. The data is collected once every minute. Field
% 4 contains temperature data.

% Channel ID to read data from
readChannelID = 12397;
% Temperature Field ID
TemperatureFieldID = 4;

% Channel Read API Key
% If your channel is private, then enter the read API
% Key between the '' below:
```

```
readAPIKey = '';

% Get temperature data from field 4 for the last 10 hours = 10 x 60
% minutes. Learn more about the THINGSPEAKREAD function by going to
% the Documentation tab on the right side pane of this page.

tempF = thingSpeakRead(readChannelID, 'Fields', TemperatureFieldID, 'NumMinutes', 10*60,
'ReadKey', readAPIKey);

histogram(tempF);
xlabel('Temperature (F)');
ylabel('Number of Measurements for each Temperature');
title('Histogram of Temperature variation');
grid on
```

Applications:

- Different parameters like wind velocity, dew point can be calculated.

Conclusion:

MATLAB apps like Visualization and Analysis can be used to illustrate data and get comprehensive analysis.

References:

[1] https://thingspeak.com/apps/matlab_visualizations/89649/edit

Experiment 29: Store DHT11 sensor data to Gdocs using Adruino Ethernet Shield

Aim:

To store DHT11 sensor data to Google Docs using Arduino and Ethernet Shield.

Objective:

To use DHT11 sensor with Arduino Ethernet shield to log the sensor data on the internet.

Components Required:

1. Arduino UNO
2. Arduino Ethernet Shield
3. Power cable
4. DHT11 Sensor module
5. Jumper Wires

Connections:

Arduino Pins	DHT11 Sensor module
Vcc	Vcc
Gnd	Gnd
D2	Out

1. Perform the following connections as shown in the table.

Procedure:

1. Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows to read values from DHT11 sensor.
2. Go to link provided in Reference [1] and download the library. Follow the instructions in their notes section if you're facing any problems.
3. Upload the demo sketch to Arduino and check if your DHT11 sensor is giving proper values.
4. Next, sign in your Google account and create a new form. Select blank form and give it title, Arduino Sensor Data

5. Next choose the question type as *Short answer type field.* In the question filed just type Humidity.
6. Insert one more question of the same type and type Temperature in the question field.
7. View your form in the browser and enter some text in first answer section. Keep the cursor there itself and right click and select *Inspect Element.*
8. Now carefully copy the highlighted text by right clicking on it, *Copy>> Copy Element.*
9. When you paste in WordPad, you should get something like this:

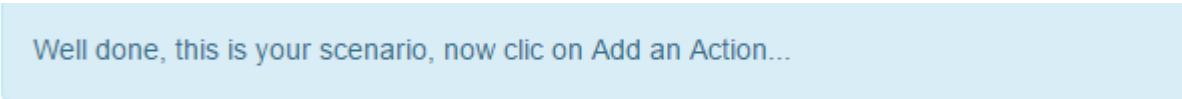
   <input type="text" class="quantumWizTextinputPaperinputInput exportInput" jsname="YPqjbf" autocomplete="off" tabindex="0" aria-label="Name of employee" aria-describedby="i.desc.619782242 i.err.619782242" name="entry.0213456257" value="" dir="auto" data-initial-dir="auto" data-initial-value="" aria-invalid="false" badinput="false">

10. We need the highlighted part shown above.
11. This is just one entry number, we need one more. So click on the other answer section and follow the same steps
12. Now submit the response and copy the url after submitting and name it "FormResponseUrl" for further reference.
    FormResponseUrl >> https://docs.google.com/forms/d/e/1FAIpQLScPbG-3JXocuK0lFY4ZpAnTDi3p28OHrSZmLqhOtDN_Cwov_Q/formResponse
13. Create account on pushing box .Link provided in Reference[3].
14. In pushing box, My Services>>Add a service>>Custom URL-'Select this Service'
15. In the pop up window, fill in these 3 options:

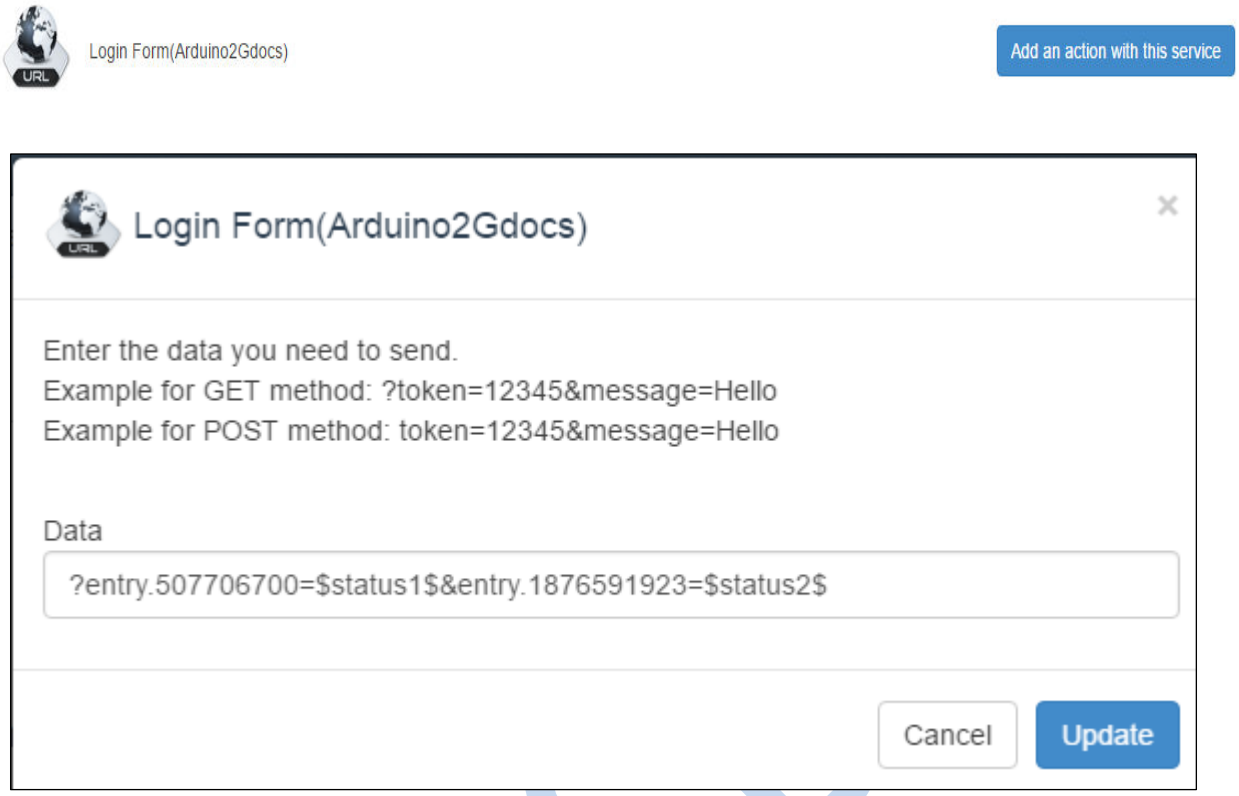
a. Name of your CustomURL configuration : Login Form(Arduino2Gdocs) (Any name of your choice, Let's stick to this for now)
b. Root URL : FormResponseUrl (From step 17)
c. Method: GET

16. Next in pushing box: Click on My Scenarios

17. As shown in image, enter any name you like. Lets go with Arduino2Gdocs. Click *Add*



18. You should get the above response. Now, click *Add an Action*

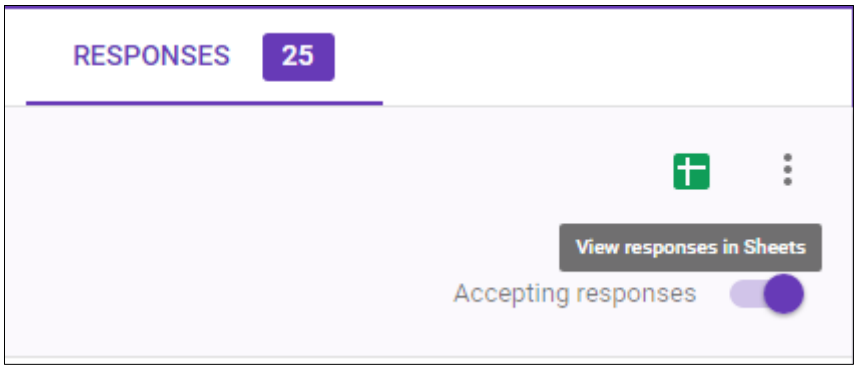


19. Click Add an action with this service where your service is Login Form(Arduino2Gdocs).In the pop up window :

20. Change the entry.xxxxxxxxxx value to your own and paste this >>
    ?entry.xxxxxxxxxx=$status1$&entry.xxxxxxxxxx=$status2$

21. One last value required before uploading code to the Arduino Board! Make note of the Device ID.
    DeviceID: vXXXXXXXXXXXXXX

22. Change the values of device ID, Rid tags in the Arduino code and Upload it to the board. Plug in the Ethernet cable to router and hover the tag on the RFID card reader to view the output.
23. Click on View responses in sheets option (green symbol) as shown in the figure below to view your output.



Hint: use the Serial Monitor for debugging purposes.

Source Code:

```
/*-----------------------------------------------
  Code by : Ms. Michelle D'Souza (IoT Intern)
  Grateful to the whole open source community on internet.
  http://playground.arduino.cc/Main/DHT11Lib
  https://github.com/Clement87/PushingBox-for-Arduino
  http://www.open-electronics.org/how-send-data-from-arduino-to-google-
docs-spreadsheet/
  Since Arduino can't handle https, we need to use Pushingbox API (uses
http) to run the Google Script (uses https).
  http://jarkomdityaz.appspot.com/
*/

#include <SPI.h>
#include <Ethernet.h>
#include <dht11.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //Setting MAC Address
char server[] = "api.pushingbox.com"; //pushingbox API server

EthernetClient client; //define 'client' as object
dht11 DHT11;

String data; //GET query with data
float data1;
float data2;
boolean ifconnected = false;

#define DHT11PIN 2

void setup() {

  Serial.begin(9600);
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
  }

  printIPAddress();
  delay(1000);

  Serial.print("DHT LIBRARY VERSION: ");
  Serial.println(DHT11LIB_VERSION);
  Serial.println();
```

```
}

void loop() {
  //Read values from sensor
  Serial.println();
  int chk = DHT11.read(DHT11PIN);
  Serial.print("Read sensor: ");

  switch (chk)
  {
    case DHTLIB_OK:
      data1 = DHT11.humidity;
      data2 = DHT11.temperature;

      Serial.println("OK");
      Serial.print("Humidity(%): ");
      Serial.print(data1);
      Serial.print("  Temperature (°C): ");
      Serial.print(data2);
      break;

    default:

      Serial.println("Unknown error");
      data1 = data2 = 0;
      break;
  }

  prepareData(); //packing GET query with data

  Serial.println();
  Serial.println("connecting...");
  if (client.connect(server, 80)) {
    sendData();
    ifconnected = true; //connected = true
  }
  else {
    Serial.println("connection failed");
  }
  // loop
  Serial.println("Http Header : ");
  while (ifconnected) {
    if (client.available()) {
      char c = client.read(); //save http header to c
      Serial.print(c); //print http header to serial monitor
    }
    if (!client.connected()) {
      Serial.println();
      Serial.println("disconnecting.");
      Serial.print("Sent Successfully!");
      //Serial.println(temp); //print sent value to serial monitor
      client.stop();
      ifconnected = false;
      data = ""; //data reset
    }
  }
  delay(5000); // interval
}
```

```
void prepareData() {
  data += "";
  data += "GET /pushingbox?devid=vE0FF42472965223&stat1="; //GET request
query to pushingbox API
  data += data1;
  data += "&stat2=";
  data += data2;
  data += " HTTP/1.1";
}
void sendData() {
  Serial.println("connected");
  client.println(data);
  client.println("Host: api.pushingbox.com");
  client.println("Connection: close");
  client.println();
}
void printIPAddress()
{
  Serial.print("My IP address: ");
  for (byte thisByte = 0; thisByte < 4; thisByte++) {
    // print the value of each byte of the IP address:
    Serial.print(Ethernet.localIP()[thisByte], DEC);
    Serial.print(".");
  }

  Serial.println();
}
```

Applications:

- Multiple sensor data can be logged and stored on the internet using Arduino and Ethernet shield.

Conclusion:

- The DHT11 sensor data is stored on Gdocs with entry time and date stored on the internet.

References:

[1] http://playground.arduino.cc/Main/DHT11Lib

[2] https://github.com/Clement87/PushingBox-for-Arduino

[3] https://www.pushingbox.com/

[4] http://jarkomdityaz.appspot.com/

[5] http://www.open-electronics.org/how-send-data-from-arduino-to-google-docs-spreadsheet/

Experiment 30 :Employee entry logging to Gdocs using RFID and Arduino Ethernet Shield

Aim:

To create an employee entry login system which records the date and time using Arduino Ethernet shield and RFID.

Objective:

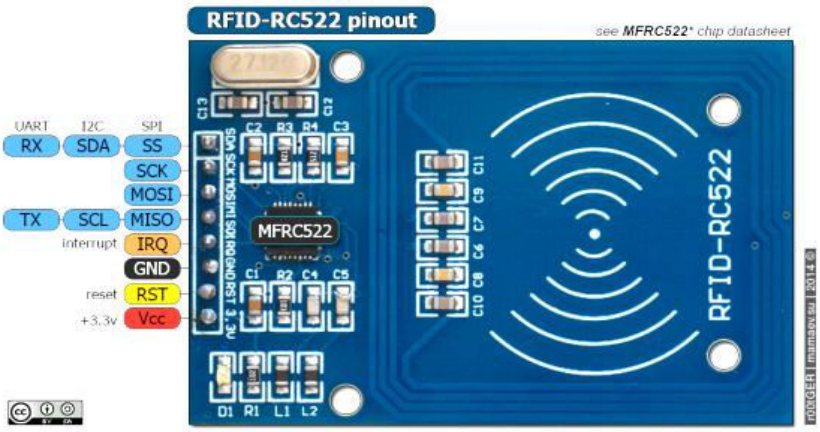
To learn about RFID and use RFID module with Arduino Ethernet shield to log the employee login time to Google spreadsheet.

Components Required:

1. Arduino UNO
2. Arduino Ethernet Shield
3. Power cable
4. RFID Chip: RC522 model
5. Jumper Wires
6. RFID Tags

Connections:

Arduino Pins	RFID-RC522
Reset	9
SPI SS	**8**
SPI MOSI	11
SPI MISO	12
SPI SCK	13



2. Perform the following connections as shown in the table.
   Note: SPI SS is connected to Pin 8 when used with Ethernet shield of Arduino.

Procedure:

24. Once you have your Adruino IDE installed and updated to the latest version, we will use an additional library which allows to read and write onto RFID cards using Arduino.
25. Go to link provided in Reference [2] and download the library.
26. Next in the Arduino IDE, click on :
27. *Sketch >> Include Library >> Add .ZIP library.*
28. Browse and select the folder where you downloaded the library.
29. Now, we have to read the RFID cards ID number in order to allow access to a specific card user only. So, in your Arduino IDE click on:
30. *File >> Examples >> MRFC522 >> ReadUidMultiReader.*
31. Plug in your Arduino with the connections made to RFID Chip and upload this code. Open the Serial Monitor. Next, place your RFID cards close to the RFID reader and note down the UID numbers for both.
32. We are good and ready to go. Create a new sketch and copy paste the code contents provided below.
33. Make sure you change the UID number according to your card in user1 and user2 in the code.
34. Now sign in your Google account and create a new form. Select blank form and give it title, Login System
35. Next choose the question type as *Short answer type field.* In the question filed just type Name of Employee.
36. View your form in the browser and enter some text. Keep the cursor there itself and right click and select *Inspect Element.*
37. Now carefully copy the highlighted text by right clicking on it, *Copy >> Copy Element.*
38. When you paste in WordPad, you should get something like this:

    <input type="text" class="quantumWizTextinputPaperinputInput exportInput" jsname="YPqjbf" autocomplete="off" tabindex="0" aria-label="Name of employee" aria-describedby="i.desc.619782242 i.err.619782242" name="entry.0213456257" value="" dir="auto" data-initial-dir="auto" data-initial-value="" aria-invalid="false" badinput="false">

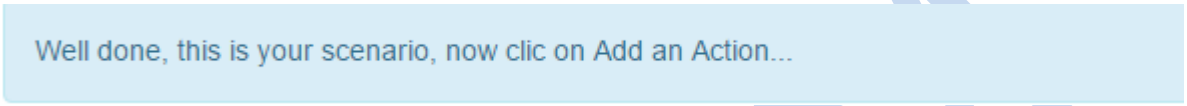
39. We need the highlighted part shown above.
40. Now submit the response and copy the url after submitting and name it “FormResponseUrl” for further reference.
    FormResponseUrl >> https://docs.google.com/forms/d/e/1FAIpQLScPbG-3JXocuK0lFY4ZpAnTDi3p28OHrSZmLqhOtDN_Cwov_Q/formResponse
41. Create account on pushing box .Link provided in reference[3].
42. In pushing box, My Services>>Add a service>>Custom URL-‘Select this Service’
43. In the pop up window, fill in these 3 options:

d. Name of your CustomURL configuration : Login Form(Arduino2Gdocs) (Any name of your choice, Let’s stick to this for now)

e. Root URL : FormResponseUrl (From step 17)
f. Method: GET

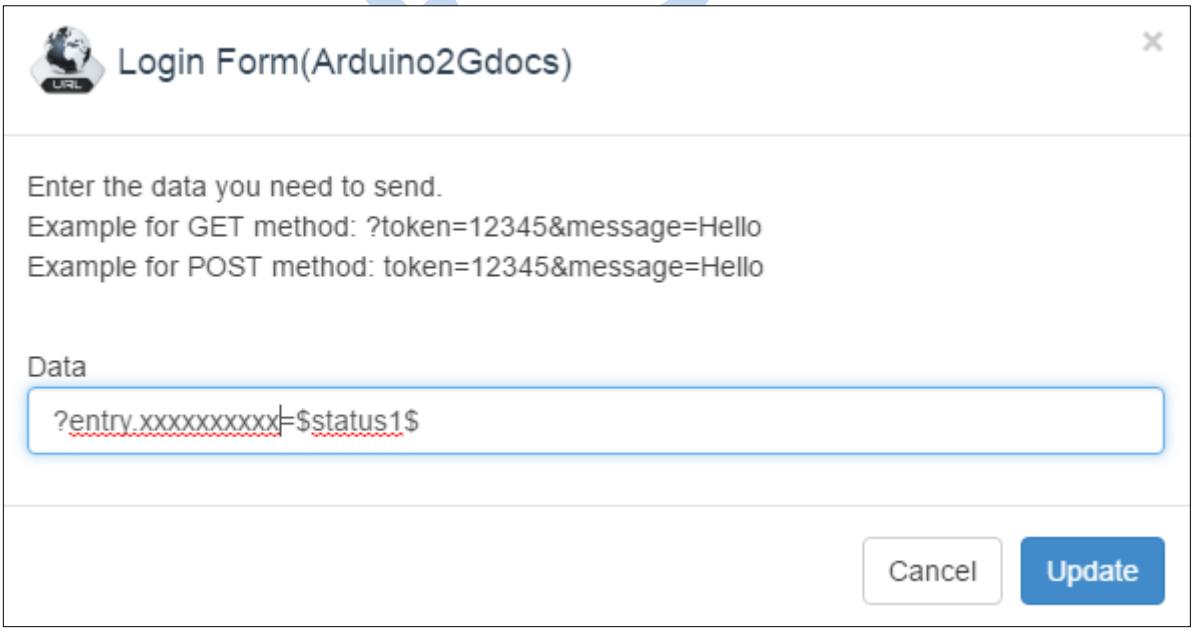
44. Next in pushing box: Click on My Scenarios

45. As shown in image, enter any name you like. Let's go with Arduino2Gdocs. Click *Add*



46. You should get the above response. Now, click *Add an Action*

47. Click Add an action with this service where your service is Login Form(Arduino2Gdocs).In the pop up window :

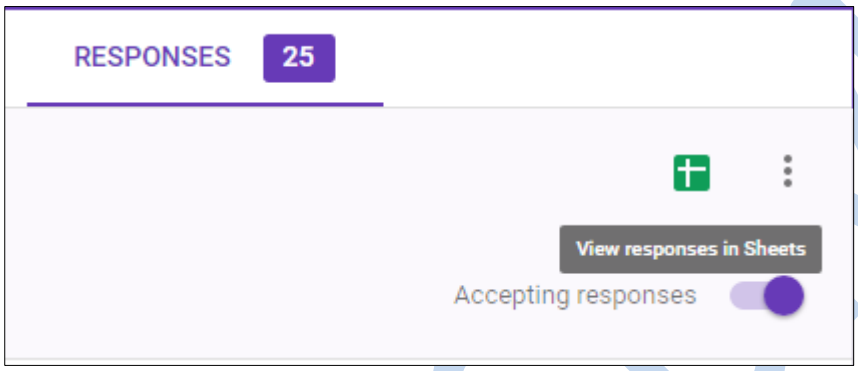


48. Change the entry.xxxxxxxxxx value to your own and paste this >>
?entry.xxxxxxxxxx=$status1$

49. One last value required before uploading code to the Arduino Board! Make note of the Device ID.

    DeviceID: vXXXXXXXXXXXXXXX

50. Change the values of device ID, Rid tags in the Arduino code and Upload it to the board. Plug in the Ethernet cable to router and hover the tag on the RFID card reader to view the output.
51. Click on View responses in sheets option (green symbol) as shown in the figure below to view your output.



Hint: use the Serial Monitor for debugging purposes.

Source Code:

```
/*-----------------------------------------------
  Code by: Ms. Michelle DSouza (IoT Intern)
  Grateful to the whole open source community on internet.
  Since Arduino can't handle https, we need to use Pushingbox API (uses http) to run the Google Script (uses https).

  http://playground.arduino.cc/Main/DHT11Lib
  https://github.com/Clement87/PushingBox-for-Arduino
  http://www.open-electronics.org/how-send-data-from-arduino-to-google-docs-spreadsheet/
  http://jarkomdityaz.appspot.com/
*/
#include <SPI.h>
#include <Ethernet.h>
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 8
#define RST_PIN 9
//-------------------------------------------------------------------------------
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //Setting MAC Address

char server[] = "api.pushingbox.com"; //pushingbox API server
EthernetClient client; //define 'client' as object
String data; //GET query with data
const String employee[2] = {"Mr.XYZ", "Mr.ABC"};
```

```
boolean ifconnected = false;
MFRC522 rfid(SS_PIN, RST_PIN); // Instance of the class
byte user1[] = {0x22, 0x22 , 0x22, 0x22};
byte user2[] = {0x22, 0x22 , 0x22, 0x22};
int led = 13;
int i;
//--------------------------------------------------------------------------------
void setup() {
 Serial.begin(9600);
 if (Ethernet.begin(mac) == 0) {
  Serial.println("Failed to configure Ethernet using DHCP");
  //Ethernet.begin(mac, ip);
 }
 delay(1000);
 printIPAddress();
 delay(1000);
 SPI.begin(); // Init SPI bus
 rfid.PCD_Init(); // Init MFRC522
 pinMode(13, OUTPUT);
}
//--------------------------------------------------------------------------------
void loop() {
 // Look for new cards
 if ( ! rfid.PICC_IsNewCardPresent())
  return;

 // Verify if the NUID has been read
 if ( ! rfid.PICC_ReadCardSerial())
  return;

 if (rfid.uid.uidByte[0] == user1[0] ||
    rfid.uid.uidByte[1] == user1[1] ||
    rfid.uid.uidByte[2] == user1[2] ||
    rfid.uid.uidByte[3] == user1[3] ) {

  Serial.println(F("Hi ! Mr. XYZ"));
  printHex(rfid.uid.uidByte, rfid.uid.size);
  //Instead of switching led on/off, a simple door open/close application can be implemented using relay.
  digitalWrite(led, HIGH);  // turn the LED on by making the voltage HIGH
  delay(1000);              // wait for a second
  digitalWrite(led, LOW);   // turn the LED off by making the voltage LOW
  // delay(1000);     // wait for a second
  i = 0;
  UploadtoGdocs();

 }

 else if (rfid.uid.uidByte[0] == user2[0] ||
       rfid.uid.uidByte[1] == user2[1] ||
       rfid.uid.uidByte[2] == user2[2] ||
       rfid.uid.uidByte[3] == user2[3] ) {

  Serial.println(F("Hi ! Mr. ABC"));
  printHex(rfid.uid.uidByte, rfid.uid.size);
  //Instead of switching led on/off, a simple door open/close application can be implemented using relay.
```

```
   digitalWrite(led, HIGH);   // turn the LED on by making the voltage HIGH
   delay(1000);               // wait for a second
   digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
   //delay(1000);             // wait for a second
   i = 1;
   UploadtoGdocs();
  }
  else {
   Serial.println(F("Sorry.Not Authorised"));
   printHex(rfid.uid.uidByte, rfid.uid.size);
  }
}

void UploadtoGdocs() {
 prepareData(); //packing GET query with data
 Serial.println("connecting...");
 if (client.connect(server, 80)) {
   sendData();
   ifconnected = true; //connected = true
 }
 else {
   Serial.println("connection failed");
 }
 // loop
 while (ifconnected) {
   if (client.available()) {
     char c = client.read(); //save http header to c
     Serial.print(c); //print http header to serial monitor
   }
   if (!client.connected()) {
     Serial.println();
     Serial.println("disconnecting.");
     Serial.print("Data Entry made for user :");
     // Serial.println(user); //print sent value to serial monitor
     client.stop();
     ifconnected = false;
     data = ""; //data reset
   }
 }
 delay(5000); // interval

}

//GET request query to pushingbox API
void prepareData() {
 data += "";
 data += "GET /pushingbox?devid=vXXXXXXXXXXXXXXXX&stat1="; //Put your device id
 data += employee[i];
 data += " HTTP/1.1";
}

void sendData() {
 Serial.println("connected");
 client.println(data);
 client.println("Host: api.pushingbox.com");
```

```
 client.println("Connection: close");
 client.println();
}

void printIPAddress()
{
 Serial.print("My IP address: ");
 for (byte thisByte = 0; thisByte < 4; thisByte++) {
   // print the value of each byte of the IP address:
   Serial.print(Ethernet.localIP()[thisByte], DEC);
   Serial.print(".");
 }
 Serial.println();
}

void printHex(byte *buffer, byte bufferSize) {
 for (byte i = 0; i < bufferSize; i++) {
   Serial.print(buffer[i] < 0x10 ? " 0" : " ");
   Serial.print(buffer[i], HEX);
 }
}
```

Applications:

- Sensor data can be logged and stored on the internet using Arduino and Ethernet shield.

Conclusion:

- RFID based entry system for employees in any organization with entry time and date stored on the internet.

References:

[1] http://playground.arduino.cc/Learning/MFRC522

[2] https://github.com/miguelbalboa/rfid

[3] https://www.pushingbox.com/

[4] http://jarkomdityaz.appspot.com/

[5] http://www.open-electronics.org/how-send-data-from-arduino-to-google-docs-spreadsheet/

[6]https://github.com/Clement87/PushingBox-for-Arduino

Experiment 31: Display DHT11 sensor data on OLED screen using the ATtiny85 Breakout Board.

Aim:

To display DHT11 sensor data on OLED Screen using Digispark ATtiny85.

Objective:

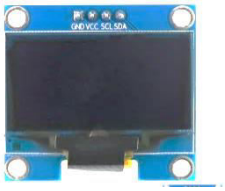
To learn about ATtiny85 microcontroller and interfacing it to DHT11 sensor, extracting the sensor data and displaying DHT11 sensor on the OLED screen.

Components Required:

1. Digispark Attiny85 USB development board
2. I2C 0.96" 128X64 White OLED Display
3. Jumper Wires

Connections:

Digispark Attiny85	OLED Display
5v	VCC
GND	GND
P2	SCL
P0	SDA



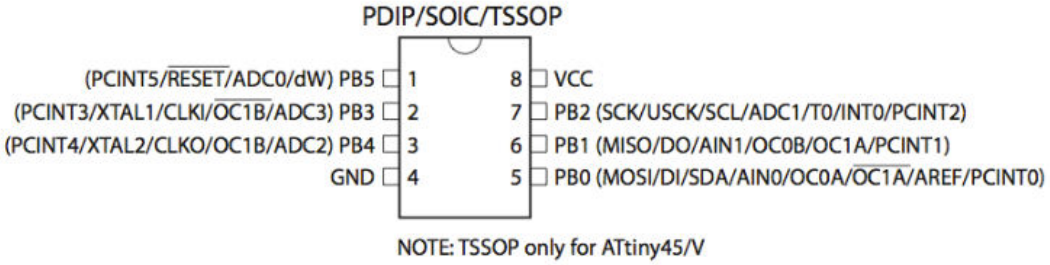
Digispark Attiny85	DHT11 Sensor
5v	VCC
GND	GND
P4	OUT

4. Perform the following connections as shown in the table.
   Note: Use the USB on the module for programming purposes only. When connecting it to any other sensor/component, use external power supply, else can damage the computer usb port.

Theory:

- ***Overview of Atiny85 chip:*** The high-performance, low-power Atmel 8-bit AVR RISC-based microcontroller combines 8KB ISP flash memory, 512B EEPROM, 512-Byte SRAM, 6 general purpose I/O lines, 32 general purpose working registers, one 8-bit timer/counter with compare modes, one 8-bit high speed timer/counter, USI, internal and external Interrupts, 4-channel 10-bit A/D converter, programmable watchdog timer with internal oscillator, three software selectable power saving modes, and debug WIRE for on-chip debugging. The device achieves a throughput of 20 MIPS at 20 MHz and operates between 2.7-5.5 volts.

Pinout ATtiny25/45/85



NOTE: TSSOP only for ATtiny45/V

- There are two types of Display available: The SPI Display and I2C Display. The SPI is faster but uses more wires for connections and thus using up more pins of the Arduino. The I2C Display uses only 2 wires for communication.

- ***Digispark USB Development Board :***
  The Digispark is an Attiny85 based microcontroller development board similar to the Arduino line, only cheaper, smaller, and a bit less powerful. With a whole host of shields to extend its functionality and the ability to use the familiar Arduino IDE the Digispark is a great way to jump into electronics, or perfect for when an Arduino is too big or too much.

- **DHT11 sensor:** This DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness.

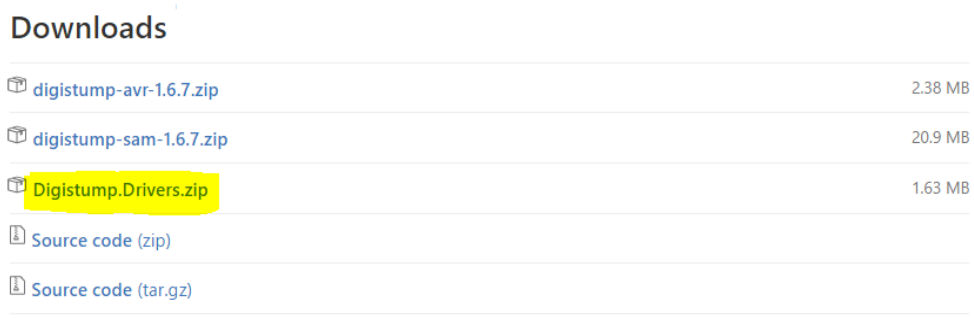
Procedure:

1. Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows us to program the ATtiny85.

2. Next in the Arduino IDE, click on:
3. *Files >> Preference >> Additional Boards Manager URLs: paste* *http://digistump.com/package_digistump_index.json*
4. Next click on

   Tools >> Board >> Boards Manager >> Type: Contributed >> Navigate to Digistump AVR Boards and install the software.
5. Perform the following step if windows does not detect and install the driver packages automatically:

   Go to link provided in Reference [3] and download and install the packages manually.



6. With this done, you are ready to program the ATtiny85 via the Arduino IDE.
7. Next we need to install packages so that we can allow it to communicate to the OLED display.
8. Go to link provided in Reference [4] and Reference [5] download the TinySSD1306 library and Adafruit DHT library. Copy and paste that folder into the libraries folder of Arduino.
9. Next, open the Arduino IDE and copy paste the code given below.
10. Select the Board as Digispark Default 16.5MHz and compile the code. Next click on upload.

Note: It will indicate when to plug in the board. No need of selecting COM PORT.

11. After programing the Board, perform the connections to the oled display and DHT11 module and connect it to external power supply of 5v.

Source Code:

```
/**
   Display DHT11 sensor value on OLED using ATtiny85.
   Code modified by : Ms. Michelle D'Souza, IoT Intern, 2016.
   Grateful to the whole open source community on internet.
*/

#include <TinySSD1306.h>
#include "DHT.h"

#define DHTPIN 4     // what digital pin we're connected to
#define DHTTYPE DHT11   // DHT 11
```

```
DHT dht(DHTPIN, DHTTYPE);

char buff[5];

void setup()
{
  display.ssd1306_begin();  // initialize with the I2C addr 0x3C (for the
128x64)
  dht.begin();
  delay(2000);
}

void loop()
{
  float hu = dht.readHumidity();
  float t = dht.readTemperature();
  // float fa = dht.toFahrenheit(te);
  display.ssd1306_str_small(15, 0, "DHT11 & ATtiny85");

  display.ssd1306_str_small(15, 3, "HUMID(%)");
  dtostrf(hu, 3, 2, buff);
  display.ssd1306_str_small(85,3, buff );

  display.ssd1306_str_small(15, 5, "TEMP(C)");
  dtostrf(t, 3, 2, buff);
  display.ssd1306_str_small(85, 5, buff );

  display.ssd1306_str_small(45, 7, "(By MichelleD)");
  delay(2000);
}
```

Applications:

Sensor data display system using ATtiny85 breakout board.

Conclusion:

Basics of the ATtiny85 chip and interfacing it to the DHT11 sensor and OLED I2C display module is accomplished and also sensor data is displayed on it.

References:

[1] http://www.atmel.com/devices/attiny85.aspx

[2] https://digistump.com/wiki/digispark/tutorials/connecting

[3] https://github.com/digistump/DigistumpArduino/releases

[4] https://drive.google.com/drive/folders/0BzjoxIwEMrM-Wk9lbkw0eVVhLUk

[5] https://github.com/adafruit/DHT-sensor-library

[6] https://www.youtube.com/watch?v=MmDBvgrYGZs

[7] http://www.micropik.com/PDF/dht11.pdf

Experiment 32: Obstacle alert system using Digispark ATtiny85

Aim:

To develop a standalone Obstacle alert system using Digispark ATtiny85.

Objective:

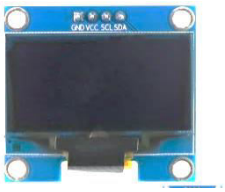
To learn about ATtiny85 microcontroller and interfacing it to IR obstacle sensor and displaying alert on OLED screen.

0**Components Required:**

1. Digispark Attiny85 USB development board
2. I2C 0.96" 128X64 White OLED Display
3. IR obstacle sensor
4. Jumper Wires

Connections:

Digispark Attiny85	OLED Display
5v	VCC
GND	GND
P2	SCL
P0	SDA



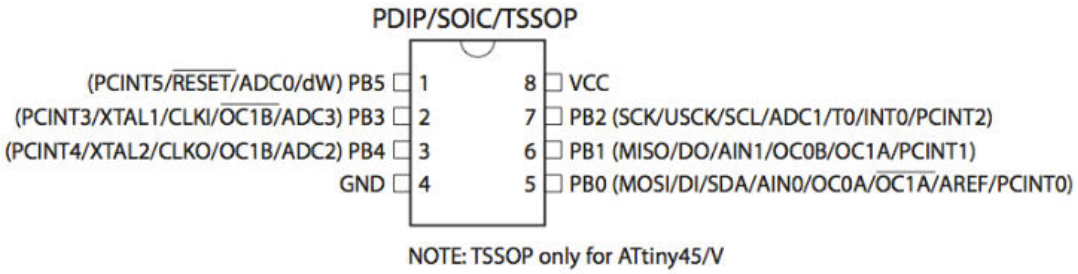
Digispark Attiny85	IR Obstacle sensor
5v	VCC
GND	GND
P4	OUT

5. Perform the following connections as shown in the table.
   Note: Use the USB on the module for programming purposes only. When connecting it to any other sensor/component, use external power supply, else can damage the computer usb port.

Theory:

- ***Overview of Atiny85 chip:*** The high-performance, low-power Atmel 8-bit AVR RISC-based microcontroller combines 8KB ISP flash memory, 512B EEPROM, 512-Byte SRAM, 6 general purpose I/O lines, 32 general purpose working registers, one 8-bit timer/counter with compare modes, one 8-bit high speed timer/counter, USI, internal and external Interrupts, 4-channel 10-bit A/D converter, programmable watchdog timer with internal oscillator, three software selectable power saving modes, and debug WIRE for on-chip debugging. The device achieves a throughput of 20 MIPS at 20 MHz and operates between 2.7-5.5 volts.

Pinout ATtiny25/45/85



NOTE: TSSOP only for ATtiny45/V

- There are two types of Display available: The SPI Display and I2C Display. The SPI is faster but uses more wires for connections and thus using up more pins of the Arduino. The I2C Display uses only 2 wires for communication.

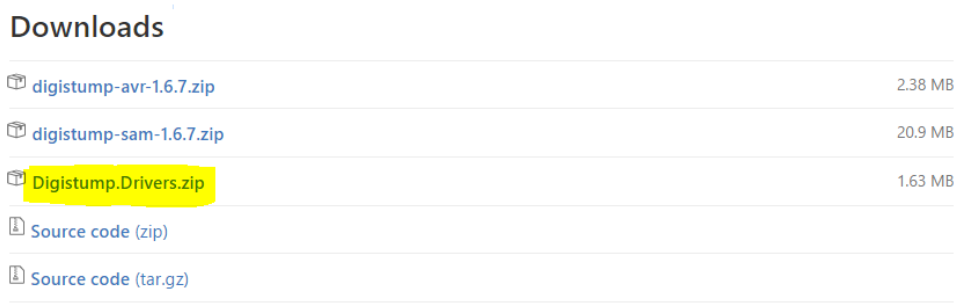
- ***Digispark USB Development Board :***
  The Digispark is an Attiny85 based microcontroller development board similar to the Arduino line, only cheaper, smaller, and a bit less powerful. With a whole host of shields to extend its functionality and the ability to use the familiar Arduino IDE the Digispark is a great way to jump into electronics, or perfect for when an Arduino is too big or too much.

Procedure:

12. Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows us to program the ATtiny85.
13. Next in the Arduino IDE, click on:
14. *Files >> Preference >> Additional Boards Manager URLs: paste* *http://digistump.com/package_digistump_index.json*
15. Next click on

    Tools >> Board >> Boards Manager >> Type: Contributed >> Navigate to Digistump AVR Boards and install the software.
16. Perform the following step if windows does not detect and install the driver packages automatically:

    Go to link provided in Reference [3] and download and install the packages manually.



17. With this done, you are ready to program the ATtiny85 via the Arduino IDE.
18. Next we need to install packages so that we can allow it to communicate to the OLED display.
19. Go to link provided in Reference [4] and download the TinySSD1306 library. Copy and paste that folder into the libraries folder of Arduino.
20. Next, open the Arduino IDE and copy paste the code given below.
21. Select the Board as Digispark Default 16.5MHz and compile the code. Next click on upload.

Note: It will indicate you when to plug in the board. No need of selecting COM PORT.

22. After programing the Board, perform the connections to the oled display and DHT11 module and connect it to external power supply of 5v.

Source Code:

```
/**
   Obstacle Alert System using Digispark ATtiny85.
   Code modified by : Ms. Michelle D'Souza, IoT Intern, 2016.
   Grateful to the whole open source community on internet.
*/
#include <TinySSD1306.h>

int LED = 1; // Use the onboard LED
int isObstaclePin = 4;  // This is our input pin

void setup()
{
  display.ssd1306_begin();  // initialize with the I2C addr 0x3C (for the
128x64)
  //  pinMode(LED, OUTPUT);
  delay(1000);
}


void loop()
{
  int isObstacle = digitalRead(isObstaclePin);
  //obstacle

  if (isObstacle == HIGH)
  {
    digitalWrite(LED, HIGH);
```

```
    display.ssd1306_prop_str_small(7, 1, "Digispark");
    display.ssd1306_str_small(62, 1, "ATtiny85");
    display.ssd1306_str_small(10, 3, "Obstacle Status:");
    display.ssd1306_str_big(8, 5, "NOT CLEAR");
    display.ssd1306_str_small(45, 7, "(by MichelleD)");

  }

  else

  { //All Clear
    //Serial.println("clear");
    digitalWrite(LED, LOW);
    display.ssd1306_prop_str_small(7, 1, "Digispark");
    display.ssd1306_str_small(62, 1, "ATtiny85");
    display.ssd1306_str_small(10, 3, "Obstacle Status:");
    display.ssd1306_str_big(8, 5, "All Clear");
    display.ssd1306_str_small(45, 7, "(by MichelleD)");
  }

  delay(200);

}
```

Applications:

Low power and easy single task systems which can be used as an alternative to the Arduino UNO.

Conclusion:

Basics of the ATtiny85 chip and simple alert on the OLED screen is achieved..

References:

[1] http://www.atmel.com/devices/attiny85.aspx

[2] https://digistump.com/wiki/digispark/tutorials/connecting

[3] https://github.com/digistump/DigistumpArduino/releases

[4] https://drive.google.com/drive/folders/0BzjoxIwEMrM-Wk9lbkw0eVVhLUk

Experiment 33: Digispark ATtiny85 and OLED screen

Aim:

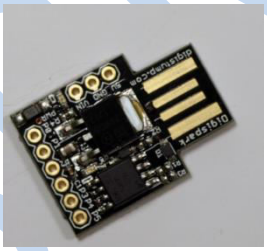
To display information on OLED Screen using Digispark ATtiny85.

Objective:

To learn about ATtiny85 microcontroller and method to program it using Arduino IDE. Also interfacing it to the OLED screen display.

Components Required:

1. Digispark Attiny85 USB development board
2. I2C 0.96" 128X64 White OLED Display
3. Jumper Wires



Connections:

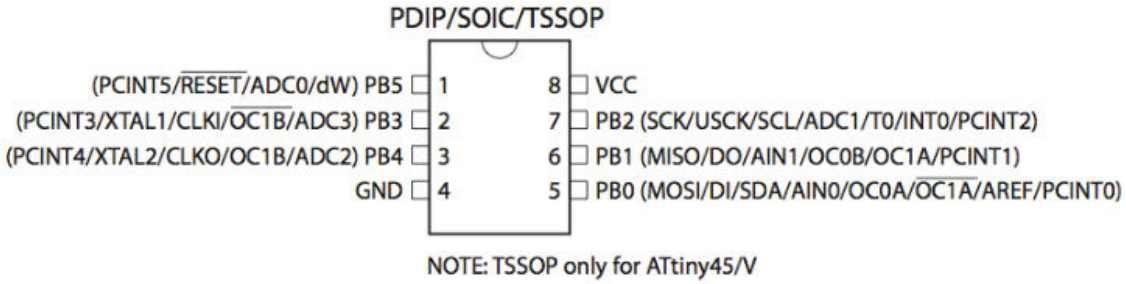
Digispark Attiny85	OLED Display
5v	VCC
GND	GND
P2	SCL
P0	SDA

6. Perform the following connections as shown in the table.
   Note: Use the USB on the module for programming purposes only. When connecting it to any other sensor/component, use external power supply ,else can damage the computer usb port.

Theory:

- ***Overview of Atiny85 chip:*** The high-performance, low-power Atmel 8-bit AVR RISC-based microcontroller combines 8KB ISP flash memory, 512B EEPROM, 512-Byte SRAM, 6 general purpose I/O lines, 32 general purpose working registers, one 8-bit timer/counter with compare modes, one 8-bit high speed timer/counter, USI, internal and external Interrupts, 4-channel 10-bit A/D converter, programmable watchdog timer with internal oscillator, three software selectable power saving modes, and debug WIRE for on-chip debugging. The device achieves a throughput of 20 MIPS at 20 MHz and operates between 2.7-5.5 volts.

Pinout ATtiny25/45/85



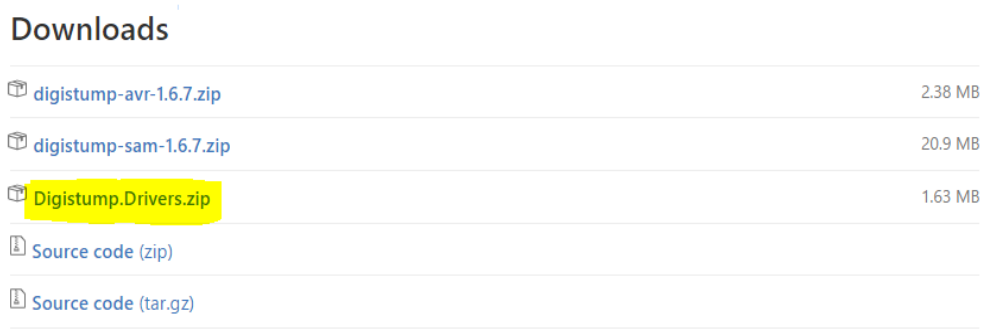
NOTE: TSSOP only for ATtiny45/V

- There are two types of Display available: The SPI Display and I2C Display. The SPI is faster but uses more wires for connections and thus using up more pins of the Arduino. The I2C Display uses only 2 wires for communication.

- ***Digispark USB Development Board :***
  The Digispark is an Attiny85 based microcontroller development board similar to the Arduino line, only cheaper, smaller, and a bit less powerful. With a whole host of shields to extend its functionality and the ability to use the familiar Arduino IDE the Digispark is a great way to jump into electronics, or perfect for when an Arduino is too big or too much.

Procedure:

23. Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows us to program the ATtiny85.
24. Next in the Arduino IDE, click on:
25. Files >> Preference >> Additional Boards Manager URLs: paste
    http://digistump.com/package_digistump_index.json
26. Next click on
    Tools >> Board >> Boards Manager >> Type: Contributed >> Navigate to Digistump AVR Boards and install the software.
27. Perform the following step if windows does not detect and install the driver packages automatically:
    Go to link provided in Reference [3] and download and install the packages manually.



28. With this done, you are ready to program the ATtiny85 via the Arduino IDE.
29. Next we need to install packages so that we can allow it to communicate to the OLED display.
30. Go to link provided in Reference [4] and download the TinySSD1306 library. Copy and paste that folder into the libraries folder of Arduino.
31. Next, open the Arduino IDE and go :
    File >> Examples >TinySSD1306 >> ssd1306_demo
32. Select the Board as Digispark Default 16.5MHz and compile the code. Next click on upload.
33. It will indicate you to plug in the board. After programing the Board, perform the connections to the oled display and connect it to external power supply of 5v.

Source Code:

```
#include <TinySSD1306.h>
#include <Wire.h>

uint8_t const PROGMEM eisenbahn_bmp[] =
{ 0x08,0xFC,0x0C,0x0C,0x0C,0xFE,0x0F,0x0F,0x0F,0xFE,
  0x0C,0x0C,0x0C,0xFE,0x0F,0x0F,0x0F,0xFE,0x0C,0x0C,
  0x0C,0xFC,0x08,0x00,0x08,0xFC,0x0C,0x0C,0x0C,0xFE,
  0x0F,0x0F,0x0F,0xFE,0x0C,0x0C,0x0C,0xFE,0x0F,0x0F,
  0x0F,0xFE,0x0C,0x0C,0x0C,0xFC,0x08,0x00,0x08,0xFC,
  0x86,0x86,0x86,0xFE,0x86,0x86,0xFE,0x84,0x80,0xF0,
  0xF8,0xF0,0x80,0x80,0x83,0xFF,0xFF,0x83,0x00,0x00,
  0x38,0x2F,0x76,0xFA,0xDA,0xFB,0x76,0x2E,0x3E,0x3F,
  0x36,0x36,0x36,0x3F,0x3E,0x2E,0x76,0xFB,0xDA,0xFA,
  0x76,0x2F,0x38,0x0C,0x38,0x2F,0x76,0xFA,0xDA,0xFB,
  0x76,0x2E,0x3E,0x3F,0x36,0x36,0x36,0x3F,0x3E,0x2E,
  0x76,0xFB,0xDA,0xFA,0x76,0x2F,0x38,0x0C,0x08,0x3F,
  0x77,0xFB,0xDB,0xFB,0x77,0x2F,0x3D,0x3D,0x35,0x35,
  0x3F,0x3D,0x3D,0x2F,0x77,0xFB,0xDB,0xFB,0x77,0x3C
};

uint8_t const PROGMEM rauch_bmp[] =
{ 0x1D,0x3B,0x7B,0x7F,0x67,0x6E,0xFC,0xF0
};

uint8_t const PROGMEM tanne_bmp[] =
{ 0x00,0x00,0x00,0x80,0xC0,0xE8,0xFC,0xFE,
```

```
  0xFF,0xFE,0xFC,0xE8,0xC0,0x80,0x00,0x00,
  0x00,0x10,0x19,0x1D,0x1F,0x1F,0x1F,0x7F,
  0x7F,0x7F,0x1F,0x1F,0x1F,0x1D,0x19,0x10
};

uint8_t const PROGMEM star_bmp[] =
{ 0x40,0xC0,0xC0,0xE0,0xE0,0xF0,0xFC,0xFF,
  0xFC,0xF0,0xE0,0xE0,0xC0,0xC0,0x40,0x00,
  0x00,0x00,0x01,0x71,0x3F,0x1F,0x0F,0x07,
  0x0F,0x1F,0x3F,0x71,0x01,0x00,0x00,0x00
};
uint8_t const PROGMEM square_lo_bmp[] =
{ 0x00,0x00,0xFC,0xFC,0xFC,0xFC,0x3C,0x3C
};

uint8_t const PROGMEM square_ro_bmp[] =
{ 0x3C,0x3C,0xFC,0xFC,0xFC,0xFC,0x00,0x00
};

uint8_t const PROGMEM square_lu_bmp[] =
{ 0x00,0x00,0x3F,0x3F,0x3F,0x3F,0x3C,0x3C
};

uint8_t const PROGMEM square_ru_bmp[] =
{ 0x3C,0x3C,0x3F,0x3F,0x3F,0x3F,0x00,0x00
};

uint8_t const PROGMEM square_lo_r_bmp[] =
{ 0x00,0x00,0xC0,0xF0,0xF8,0xF8,0x7C,0x3C
};

uint8_t const PROGMEM square_ro_r_bmp[] =
{ 0x3C,0x7C,0xF8,0xF8,0xF0,0xC0,0x00,0x00
};

uint8_t const PROGMEM square_lu_r_bmp[] =
{ 0x00,0x00,0x03,0x0F,0x1F,0x1F,0x3E,0x3C
};

uint8_t const PROGMEM square_ru_r_bmp[] =
{ 0x3C,0x3E,0x1F,0x1F,0x0F,0x03,0x00,0x00
};

uint8_t const PROGMEM square_v_bmp[] =
{ 0x00,0x00,0xFF,0xFF,0xFF,0xFF,0x00,0x00
};

uint8_t const PROGMEM square_h_bmp[] =
{ 0x3C,0x3C,0x3C,0x3C,0x3C,0x3C,0x3C,0x3C
};


void setup()   {
  display.ssd1306_begin();  // initialize with the I2C addr 0x3C (for the
128x64)
  randomSeed(analogRead(0));
}

void loop() {
```

```
  textoutput();
  blinking_stars();
  draw_rectangles();
  xmas_greetings();
}

void blinking_stars() {
  unsigned long startTime = millis();
  uint8_t x,y,z;

  display.ssd1306_fill_area();
  while ((unsigned long)(millis() - startTime) <= 25000) {
    x = random(8);
    y = random(4);
    z = random(2);
    if (z)
      display.ssd1306_draw_bmp(x*16,y*2,16,2,star_bmp);
    else
      display.ssd1306_fill_area(0,x*16,y*2,16,2);
    delay(100);
  }
  display.ssd1306_inverted(true);
  display.ssd1306_startscroll(LEFT_UP);
  delay(5000);
  display.ssd1306_stopscroll();
  display.ssd1306_inverted(false);
}

void xmas_greetings() {
  display.ssd1306_fill_area();

  display.ssd1306_draw_bmp(4,0,16,2,tanne_bmp);
  display.ssd1306_draw_bmp(27,0,16,2,tanne_bmp);
  display.ssd1306_draw_bmp(85,0,16,2,tanne_bmp);
  display.ssd1306_draw_bmp(108,0,16,2,tanne_bmp);

  display.ssd1306_str_small(48,0,"Merry");
  display.ssd1306_str_small(52,1,"XMAS");

  display.ssd1306_draw_bmp(2,2,8,1,rauch_bmp);
  display.ssd1306_draw_bmp(10,3,8,1,rauch_bmp);
  display.ssd1306_draw_bmp(42,3,8,1,rauch_bmp);
  display.ssd1306_draw_bmp(50,4,8,1,rauch_bmp);
  display.ssd1306_draw_bmp(66,2,8,1,rauch_bmp);
  display.ssd1306_draw_bmp(74,3,8,1,rauch_bmp);
  display.ssd1306_draw_bmp(114,4,8,1,rauch_bmp);
  display.ssd1306_draw_bmp(122,5,8,1,rauch_bmp);

  display.ssd1306_draw_bmp(0,6,70,2,eisenbahn_bmp);

  display.ssd1306_startscroll(RIGHT_UP,6,7,4,2,4,1);
  delay(23100);
  display.ssd1306_startscroll(RIGHT_UP,6,1,4,2,4,1);
  delay(15000);
  display.ssd1306_stopscroll();
}

void rectangle(uint8_t x, uint8_t y, uint8_t dx, uint8_t dy, boolean roundc
= false, boolean invers = false) {
```

```
  uint8_t ix,iy;
  if (roundc) {
    display.ssd1306_draw_bmp(x*8,y,8,1,square_lo_r_bmp, invers);
    display.ssd1306_draw_bmp(x*8,y+dy-1,8,1,square_lu_r_bmp, invers);
    display.ssd1306_draw_bmp((x+dx-1)*8,y,8,1,square_ro_r_bmp, invers);
    display.ssd1306_draw_bmp((x+dx-1)*8,y+dy-1,8,1,square_ru_r_bmp,
invers);
  } else {
    display.ssd1306_draw_bmp(x*8,y,8,1,square_lo_bmp, invers);
    display.ssd1306_draw_bmp(x*8,y+dy-1,8,1,square_lu_bmp, invers);
    display.ssd1306_draw_bmp((x+dx-1)*8,y,8,1,square_ro_bmp, invers);
    display.ssd1306_draw_bmp((x+dx-1)*8,y+dy-1,8,1,square_ru_bmp, invers);
  }
  for (ix=x+1; ix<x+dx-1; ix++) {
    display.ssd1306_draw_bmp(ix*8,y,8,1,square_h_bmp, invers);
    display.ssd1306_draw_bmp(ix*8,y+dy-1,8,1,square_h_bmp, invers);
  }
  for (iy=y+1; iy<y+dy-1; iy++) {
    display.ssd1306_draw_bmp(x*8,iy,8,1,square_v_bmp, invers);
    display.ssd1306_draw_bmp((x+dx-1)*8,iy,8,1,square_v_bmp, invers);
  }
}

void textoutput() {
  display.ssd1306_fill_area();
  rectangle(0,0,16,8,true);
  display.ssd1306_str_big(42, 1, "DEMO");
  display.ssd1306_prop_str_small(11,2,"OLED",true);
  display.ssd1306_prop_str_small(94,2,"0.96",true);
  display.ssd1306_prop_str_small(13,3,"SSD1306");
  display.ssd1306_str_small(62,3,"Digistump");
  display.ssd1306_prop_str_small(10,5,"proportional ein");
  display.ssd1306_str_small(10,6,"proportional aus");
  delay(20000);
}

void draw_rectangles() {
  unsigned long startTime;
  uint8_t x,y,dx,dy,z;

  for (x=0; x<20; x++) {
    rectangle(0,0,16,8,false, false);
    rectangle(1,1,14,6,false, true);
    rectangle(2,2,12,4,false, false);
    rectangle(3,3,10,2,false, true);
    rectangle(0,0,16,8,true, true);
    rectangle(1,1,14,6,true, false);
    rectangle(2,2,12,4,true, true);
    rectangle(3,3,10,2,true, false);
  }

  display.ssd1306_fill_area();
  startTime = millis();
  while ((unsigned long)(millis() - startTime) <= 20000) {
    x = random(15);
    dx = random(17-x);
    dx = max(2,dx);
    y = random(7);
    dy = random(9-y);
```

```
    dy = max(2,dy);
    z = random(2);
    rectangle(x,y,dx,dy,(z==1));
    delay(50);
  }
}
```

Applications:

Sensor data display system using ATtiny85 breakout board.

Conclusion:

Basics of the ATtiny85 chip and interfacing it to the OLED I2C display module is accomplished.

References:

[1] http://www.atmel.com/devices/attiny85.aspx

[2] https://digistump.com/wiki/digispark/tutorials/connecting

[3] https://github.com/digistump/DigistumpArduino/releases

[4] https://drive.google.com/drive/folders/0BzjoxIwEMrM-Wk9lbkw0eVVhLUk

[5] https://www.youtube.com/watch?v=MmDBvgrYGZs

Experiment 34: Led control on Esp8266 programmed via Lua.

Aim:

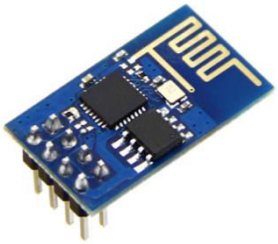
To program the ESP8266 to blink an LED.

Objective:

To learn basics of LUA programming language and code the ESP to blink an LED using Lua Script.

Components Required:

1. ESP8266 WiFi Module
2. Resistor
3. Led
4. FTDI Basic Breakout
4. Connecting wires

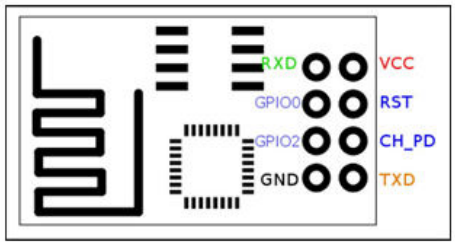


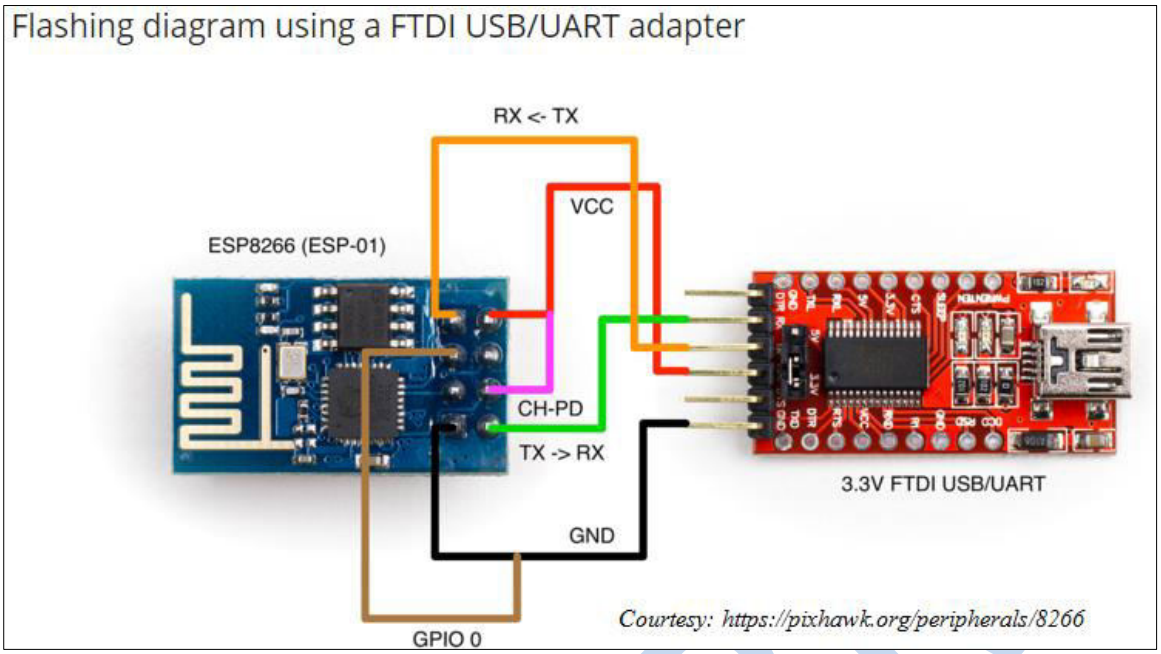
Connections:

Connections of the ESP8266 while flashing firmware:

ESP8266 Pins	FTDI
GND	GND
RX	TX
VCC	3v3
CH_PD	3v3
TX	RX
GPIO0	GND

Pinout of ESP8266:





Theory:

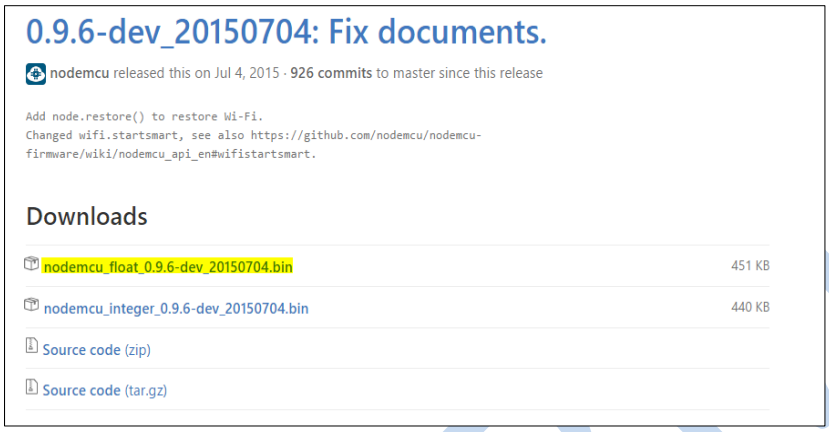
ESP8266: The ESP8266 integrates a 160 MHz microcontroller with a full Wi-Fi front-end (both as client and access point) and TCP/IP stack with DNS. ESP8266's flexibility and price point have driven its explosion of use. It is produced by Shanghai-based Chinese manufacturer, Espressif. The chip became popular for IoT with the ESP-01 module, made by a third-party manufacturer, AI-Thinker.

Engineered for mobile devices, wearable electronics and the Internet of Things (IoT) applications, ESP8266EX achieves low power consumption with a combination of several proprietary technologies. The power saving architecture features three modes of operation – active mode, sleep mode and deep sleep mode, thus allowing battery-powered designs to run longer.

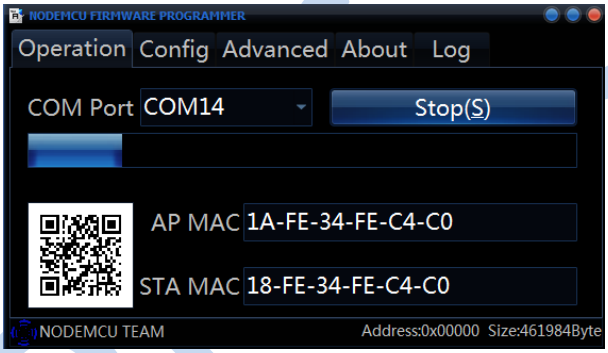
Lua: A powerful, efficient, lightweight, embeddable scripting language. It supports procedural programming, object-oriented programming, functional programming, data-driven programming, and data description.

Procedure:

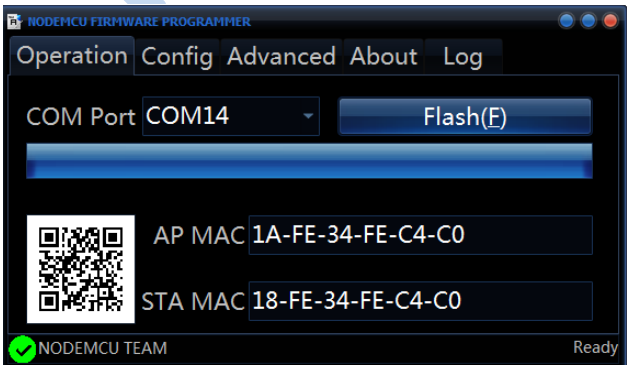
- The NodeMCU Flasher is a Firmware Programer which will allow us to install our required firmware onto the esp8266 chip. Go to link provided in Reference [1] and download and install the Flasher.
- Next go to link provided in Reference [2] and download the binary file as shown in the image which we need to flash on the esp8266.



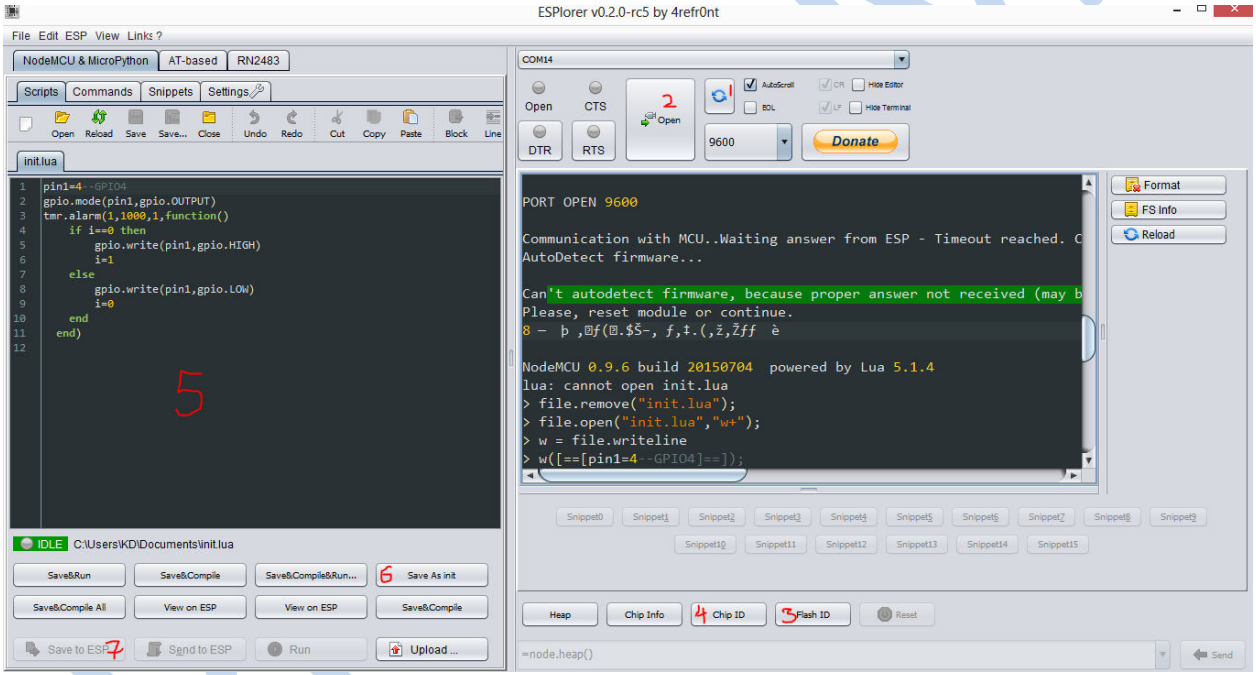
- Open the NodeMCU Flasher which you just installed. Click *Config >> Settings Icon >> Browse to the binary file >> Open >> Operations*
- Next, set up connections while flashing as shown above and plug in the USB. The COM PORT will automatically get detected. Click on *Flash.*



Note: The blue LED on the ESP should blink continuously during flashing. If this is not taking place, stop the operation and power down the Esp module. Power it back on again and Flash. If the AP MAC appears, then the flashing process is successful.



- Once flashed**, power off the ESP and also remove the connection of GPIO0 to Ground.**
- Go to link in reference [3] and download the ESPlorer software which will allow us to write Lua code on ESP.
  Caution: The ESPlorer software requires Java to be installed in order to be in the 'Executable Jar File ' format. If you're facing any issues running the software, you might want to update to the latest version of Java available.
- The image below has Numbers which will help with the programming. Plug the USB connection .Refresh (Number 1) and COM PORT will appear. Click Open (Number 2).
- *lua: cannot open init.lua* << Message may be displayed.
- Click on Flash ID (Number 3) and CHIP ID (Number 4) to see the relevant information.
- Copy Paste the code given below in Scripts section (Number 5). Save as init.lua (Number 6). Save to ESP (Number 7).



- Power down the Esp module. Connect GPIO2 to an LED and power it back on.
- By default the ESP runs the code in init.lua and your LED should be blinking.

Source Code:

```
pin1=4--GPIO4
gpio.mode(pin1,gpio.OUTPUT)
tmr.alarm(1,1000,1,function()
    if i==0 then
        gpio.write(pin1,gpio.HIGH)
        i=1
    else
        gpio.write(pin1,gpio.LOW)
        i=0
```

```
    end

  end)
```

Conclusion:

The GPIO of the ESP is used to successfully blink an Led using Lua programming language. In this way, by connecting the ESP to internet, many IoT based standalone projects can be developed.

References:

[1] https://github.com/nodemcu/nodemcu-flasher

[2] https://github.com/nodemcu/nodemcu-firmware/releases

[3] http://esp8266.ru/esplorer/

[4] https://nodemcu.readthedocs.io/en/master/en/modules/gpio/#gpioread

[5] https://www.youtube.com/watch?v=XwHfKsz1xW4

[6] https://iotbytes.wordpress.com/esp8266-2/

Experiment 35: Led control on Esp8266 programmed via Arduino IDE.

Aim:

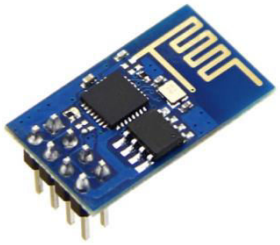
To program the ESP8266 to blink an LED using the Arduno IDE.

Objective:

To learn basics of LUA programming language and code the ESP to blink an LED using Lua Script.

Components Required:

1. ESP8266 WiFi Module
2. Resistor
3. Led
4. Connecting wires

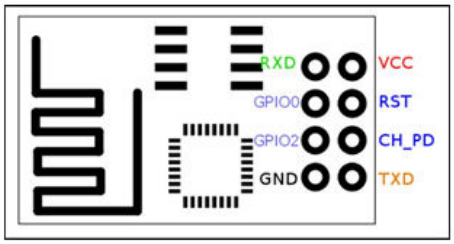


Connections:

Connections of the ESP8266:

ESP8266 Pins	FTDI Pins
GND	GND
RX	D3
VCC	3v3
CH_PD	3v3
TX	D2
GPIO0	GND
RST	GND

Pinout of ESP8266:



Theory:

ESP8266: The ESP8266 integrates a 160 MHz microcontroller with a full Wi-Fi front-end (both as client and access point) and TCP/IP stack with DNS. ESP8266's flexibility and price point have driven its explosion of use. It is produced by Shanghai-based Chinese manufacturer, Espressif. The chip became popular for IoT with the ESP-01 module, made by a third-party manufacturer, AI-Thinker.

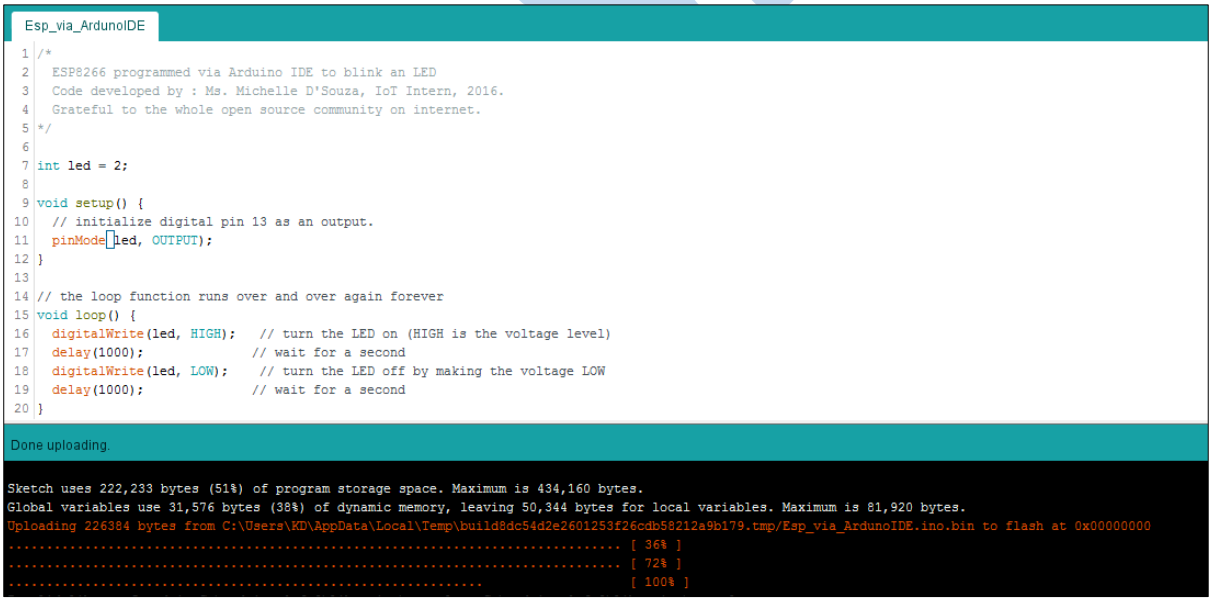
Engineered for mobile devices, wearable electronics and the Internet of Things (IoT) applications, ESP8266EX achieves low power consumption with a combination of several proprietary technologies. The power saving architecture features three modes of operation – active mode, sleep mode and deep sleep mode, thus allowing battery-powered designs to run longer.

Procedure:

34. Open the Arduino IDE and go to files and click on the preference in the Arduino IDE.
    Files >> Preference >> Additional Boards Manager URLs: paste *http://arduino.esp8266.com/stable/package_esp8266com_index.json* *>> Ok*
35. Next click on

    Tools >> Board >> Boards Manager >> search for esp8266 >> Navigate to esp8266 by Esp8266 community and install the software for Arduino.

36. Copy paste the code given below. Plug in the USB connection. Select the com port and chose Board as Generic ESP8266 Module.
37. Click on Upload and then **remove the Reset connection** to ground at this point.



38. Once uploaded, Power off the ESP module and remove the GPIO0 connection to ground .
39. Make the connection of GPIO2 to the Led and power in the module. Your led should be blinking.

Source Code:

```
/*
  ESP8266 programmed via Arduino IDE to blink an LED
  Code developed by : Ms. Michelle D'Souza, IoT Intern, 2016.
  Grateful to the whole open source community on internet.
*/

int led = 2;

void setup() {
  // initialize digital pin 13 as an output.
  pinMode(led, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage
level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the
voltage LOW
  delay(1000);               // wait for a second
}
```

Conclusion:

The GPIO of the ESP is used to successfully blink an Led using the well-known Arduino programming. In this way, by connecting the ESP to internet, many IoT based standalone projects can be developed.

References:

[1] http://www.whatimade.today/esp8266-easiest-way-to-program-so-far/

[2] http://iot-playground.com/blog/2-uncategorised/38-esp8266-and-arduino-ide-blink-example

Experiment 36: ESP8266 GPIO Control via Web Server programmed via Lua Script.

Aim:

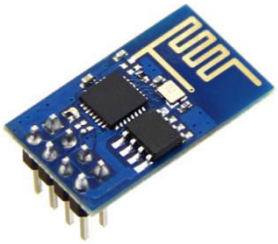
To control the GPIO Pins of the ESP8266 via a Web Browser using Lua Script.

Objective:

To learn basics of LUA programming language and code the ESP to blink an LED via a Web Server using Lua Script.

Components Required:

1. ESP8266 WiFi Module
2. Resistor
3. Led
4. FTDI Basic Breakout
4. Connecting wires

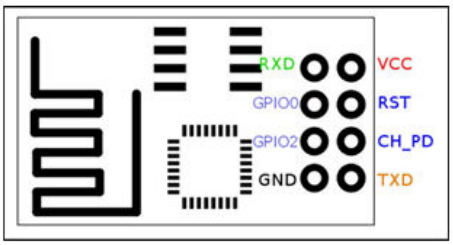


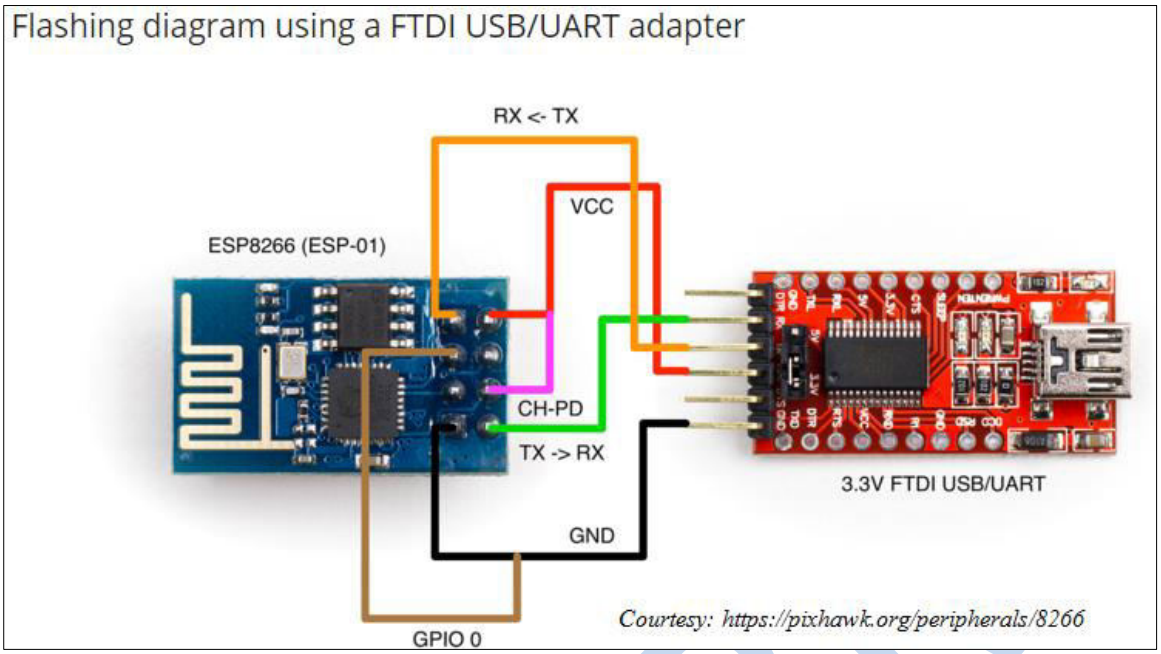
Connections:

Connections of the ESP8266 while flashing firmware:

ESP8266 Pins	FTDI
GND	GND
RX	TX
VCC	3v3
CH_PD	3v3
TX	RX
GPIO0	GND

Pinout of ESP8266:





Theory:

ESP8266:The ESP8266 integrates a 160 MHz microcontroller with a full Wi-Fi front-end (both as client and access point) and TCP/IP stack with DNS. ESP8266's flexibility and price point have driven its explosion of use. It is produced by Shanghai-based Chinese manufacturer, Espressif. The chip became popular for IoT with the ESP-01 module, made by a third-party manufacturer, AI-Thinker.

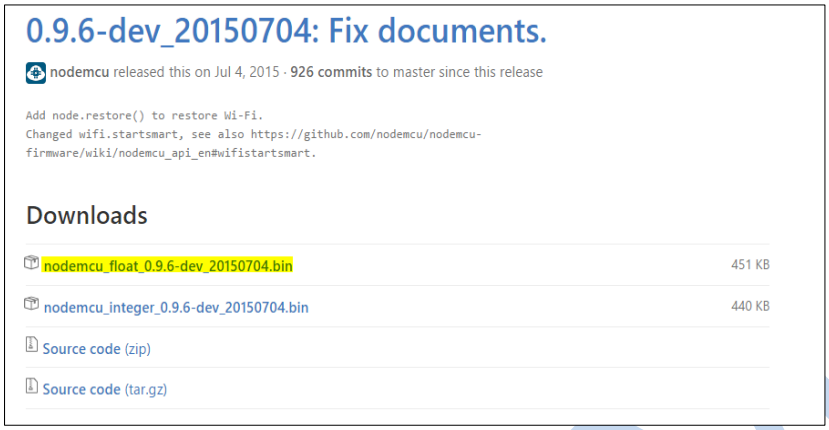
Engineered for mobile devices, wearable electronics and the Internet of Things (IoT) applications, ESP8266EX achieves low power consumption with a combination of several proprietary technologies. The power saving architecture features three modes of operation – active mode, sleep mode and deep sleep mode, thus allowing battery-powered designs to run longer.

Lua: A powerful, efficient, lightweight, embeddable scripting language. It supports procedural programming, object-oriented programming, functional programming, data-driven programming, and data description.

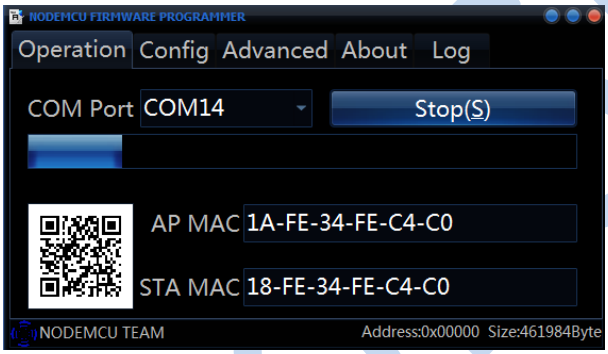
Procedure:

- The NodeMCU Flasher is a Firmware Programer which will allow us to install our required firmware onto the esp8266 chip. Go to link provided in Reference [1] and download and install the Flasher.

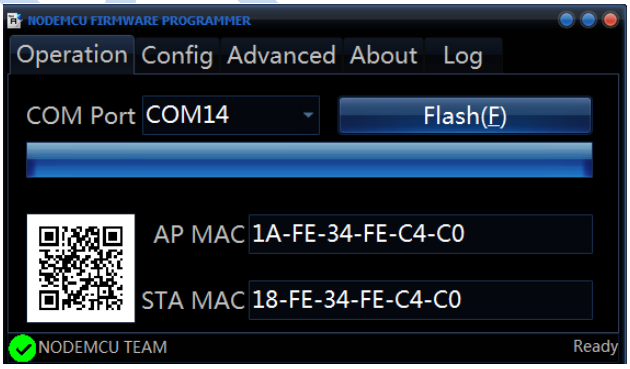
- Next go to link provided in Reference [2] and download the binary file as shown in the image which we need to flash on the esp8266.



- Open the NodeMCU Flasher which you just installed. Click *Config >> Settings Icon >> Browse to the binary file >> Open >> Operations*
- Next, set up connections while flashing as shown above and plug in the USB. The COM PORT will automatically get detected. Click on *Flash.*



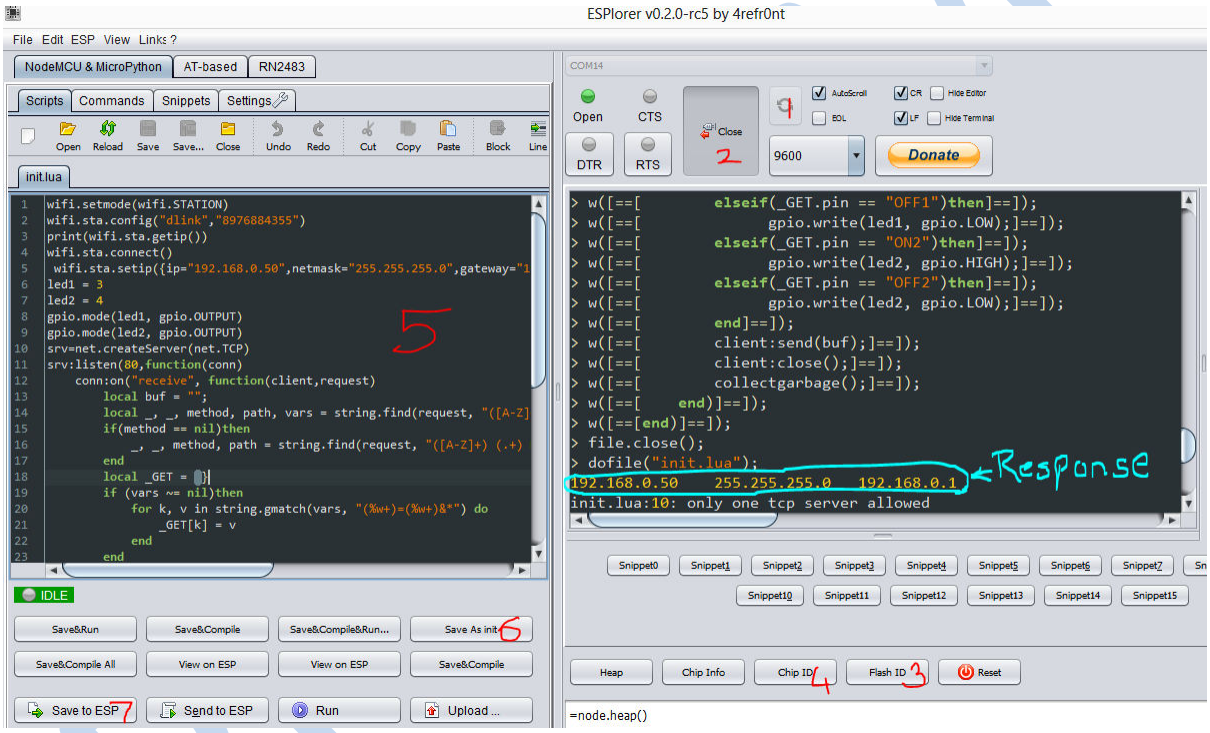
Note: The blue LED on the ESP should blink continuously during flashing. If this is not taking place, stop the operation and power down the Esp module. Power it back on again and Flash. If the AP MAC appears, then the flashing process is successful.



- Once flashed**, power off the ESP and also remove the connection of GPIO0 to Ground.**
- Go to link in reference [3] and download the ESPlorer software which will allow us to write Lua code on ESP.

Caution: The ESPlorer software requires Java to be installed in order to be in the 'Executable Jar File ' format. If you're facing any issues running the software, you might want to update to the latest version of Java available.

- The image below has Numbers which will help with the programming. Plug the USB connection .Refresh (Number 1) and COM PORT will appear. Click Open (Number 2).
- *lua: cannot open init.lua* << Message may be displayed.
- Click on Flash ID (Number 3) and CHIP ID (Number 4) to see the relevant information.
- Copy Paste the code given below in Scripts section (Number 5). Save as init.lua (Number 6). Save to ESP (Number 7).



- Power down the Esp module. Connect GPIO2 and GPIO0 to an LED each and power it back on.
- By default the ESP runs the code in init.lua file. Go to your browser and type the static IP address we gave >> 192.168.0.50.
- Click on the buttons to see the Led's turn on/off.

Source Code:

```
--[[
     ESP8266 GPIO Control via Web Server.
     Grateful to the whole open source community on internet.
     Code modified by : Ms. Michelle D'Souza, IoT Intern, 2016.
     http://www.instructables.com/id/Esp8266-Web-Control-with-Nodemcu-and-
Lua/?ALLSTEPS
    --]]

wifi.setmode(wifi.STATION)
wifi.sta.config("YOUR WIFI SSID","YOUR WIFI PASSWORD")
print(wifi.sta.getip())
wifi.sta.connect()

wifi.sta.setip({ip="192.168.0.50",netmask="255.255.255.0",gateway="192.168.
0.1"})
led1 = 3
led2 = 4
gpio.mode(led1, gpio.OUTPUT)
gpio.mode(led2, gpio.OUTPUT)
srv=net.createServer(net.TCP)
srv:listen(80,function(conn)
    conn:on("receive", function(client,request)
        local buf = "";
        local _, _, method, path, vars = string.find(request, "([A-Z]+)
(.+)?(.+) HTTP");
        if(method == nil)then
            _, _, method, path = string.find(request, "([A-Z]+) (.+)
HTTP");
        end
        local _GET = {}
        if (vars ~= nil)then
            for k, v in string.gmatch(vars, "(%w+)=(%w+)&*") do
                _GET[k] = v
            end
        end

        buf = buf.."<head><meta name='viewport' content='width=200px,
initial-scale=1.0'/>"
        buf = buf.."<font face='verdana'><h1> ESP8266 GPIO Control via Web
Server</h1>";
        buf = buf.."<h2><p>GPIO0 :<a
href=\"?pin=ON1\"> <button>ON</button></a> <a
href=\"?pin=OFF1\"><button>OFF</button></a></p>";
```

```
        buf = buf.."<p>GPIO2 : <a
href=\"?pin=ON2\"><button>ON</button></a> <a
href=\"?pin=OFF2\"><button>OFF</button></a></p></font></head>";
        local _on,_off = "",""
        if(_GET.pin == "ON1")then
              gpio.write(led1, gpio.HIGH);
        elseif(_GET.pin == "OFF1")then
              gpio.write(led1, gpio.LOW);
        elseif(_GET.pin == "ON2")then
              gpio.write(led2, gpio.HIGH);
        elseif(_GET.pin == "OFF2")then
              gpio.write(led2, gpio.LOW);
        end
        client:send(buf);
        client:close();
        collectgarbage();
    end)

end)
```

Conclusion:

The GPIO of the ESP is used to successfully blink an Led using Lua programming language via a Web Browser. In this way, many Web based IoT standalone projects can be developed.

References:

[1] https://github.com/nodemcu/nodemcu-flasher

[2] https://github.com/nodemcu/nodemcu-firmware/releases

[3] http://esp8266.ru/esplorer/

[4] http://www.nodemcu.com/index_en.html

[5] https://www.youtube.com/watch?v=48JUh8Vlea4

[6] https://www.youtube.com/watch?v=toUgzpI8HHo

[7]

http://www.instructables.com/id/Esp8266-Web-Control-with-Nodemcu-and-Lua/?ALLSTEPS

Experiment 37: ESP8266 GPIO Control via Web Server programmed via Arduino IDE.

Aim:

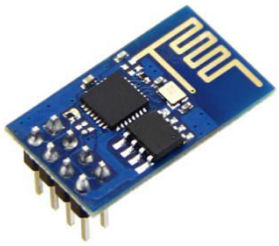
To control the GPIO pins of the ESP8266 via a Web Browser programmed using the Arduino IDE.

Objective:

To learn how to create a server and control GPIO of ESP8266 using well known Arduino programming.

Components Required:

1. ESP8266 WiFi Module
2. Resistor
3. Led
4. Connecting wires

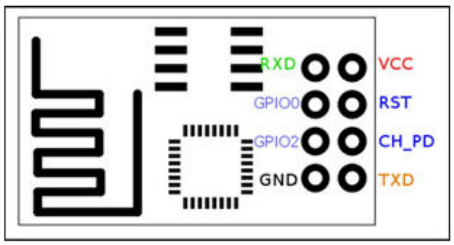


Connections:

Connections of the ESP8266:

ESP8266 Pins	FTDI Pins
GND	GND
RX	D3
VCC	3v3
CH_PD	3v3
TX	D2
GPIO0	GND
RST	GND

Pinout of ESP8266:



Theory:

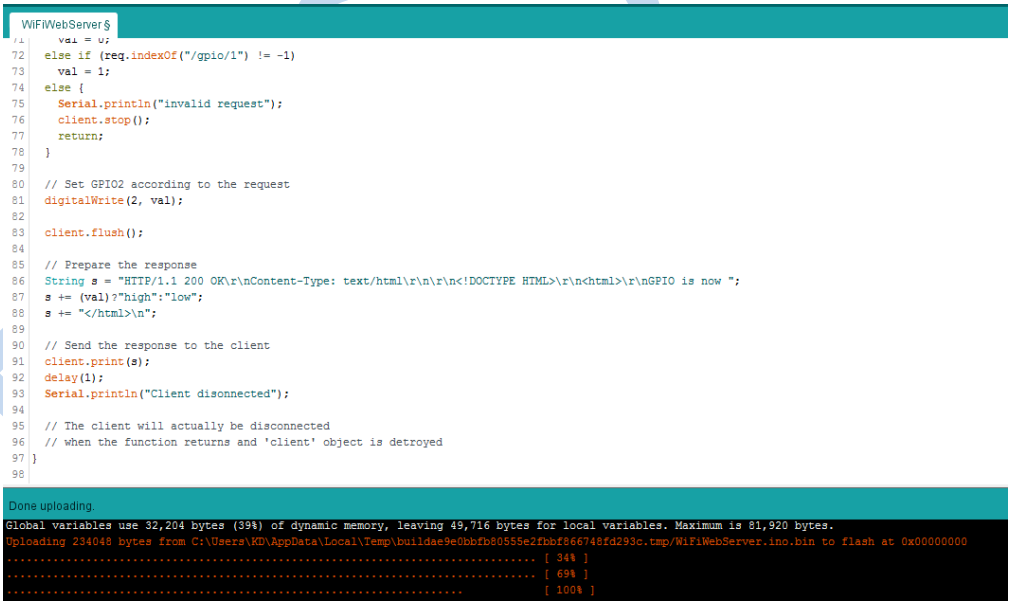
ESP8266: The ESP8266 integrates a 160 MHz microcontroller with a full Wi-Fi front-end (both as client and access point) and TCP/IP stack with DNS. ESP8266's flexibility and price point have driven its explosion of use. It is produced by Shanghai-based Chinese manufacturer, Espressif. The chip became popular for IoT with the ESP-01 module, made by a third-party manufacturer, AI-Thinker.

Engineered for mobile devices, wearable electronics and the Internet of Things (IoT) applications, ESP8266EX achieves low power consumption with a combination of several proprietary technologies. The power saving architecture features three modes of operation – active mode, sleep mode and deep sleep mode, thus allowing battery-powered designs to run longer.

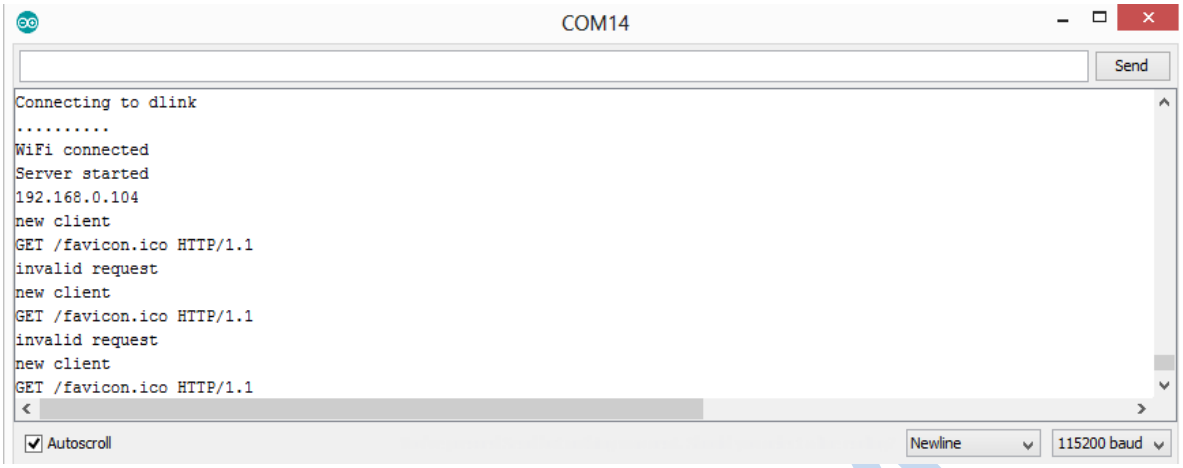
Procedure:

40. Open the Arduino IDE and go to files and click on the preference in the Arduino IDE. *Files >> Preference >> Additional Boards Manager URLs: paste* http://arduino.esp8266.com/stable/package_esp8266com_index.json *>> Ok*
41. Next click on

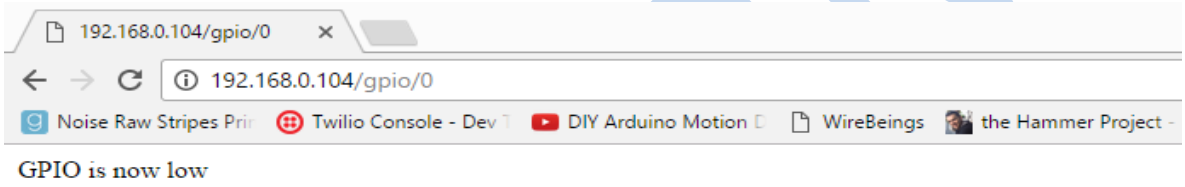
    Tools >> Board >> Boards Manager >> search for esp8266 >> Navigate to esp8266 by Esp8266 community and install the software for Arduino.
42. Click on *File >> Examples >> Esp8266WiFi >> WiFiWebServer.*
43. Put down your own wifi ssid and password.
44. Plug in the USB connection. Select the com port and chose Board as Generic ESP8266 Module.
45. Click on Upload and then **remove the Reset connection** to ground at this point.



46. Once uploaded, Power off the ESP module and remove the GPIO0 connection to ground.
47. Make the connection of GPIO2 to the Led and power on the module. Open the serial monitor of Arduino to know which local Ip address is the server established.
48. Go to your browser and type the IP address of the server according to your own >> 192.168.0.104



49. By typing >> *http://server_ip/gpio/1* will set the GPIO2 high. Your led should glow.



Source Code:

```
/*
 *  This sketch demonstrates how to set up a simple HTTP-like
server.
 *  The server will set a GPIO pin depending on the request
 *    http://server_ip/gpio/0 will set the GPIO2 low,
 *    http://server_ip/gpio/1 will set the GPIO2 high
 *  server_ip is the IP address of the ESP8266 module, will be
 *  printed to Serial when the module is connected.
 */

#include <ESP8266WiFi.h>

const char* ssid = "your-ssid";
const char* password = "your-password";

// Create an instance of the server
// specify the port to listen on as an argument
WiFiServer server(80);

void setup() {
  Serial.begin(115200);
```

```
  delay(10);

  // prepare GPIO2
  pinMode(2, OUTPUT);
  digitalWrite(2, 0);

  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the server
  server.begin();
  Serial.println("Server started");

  // Print the IP address
  Serial.println(WiFi.localIP());
}

void loop() {
  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  // Wait until the client sends some data
  Serial.println("new client");
  while(!client.available()){
    delay(1);
  }

  // Read the first line of the request
  String req = client.readStringUntil('\r');
  Serial.println(req);
  client.flush();

  // Match the request
  int val;
  if (req.indexOf("/gpio/0") != -1)
    val = 0;
  else if (req.indexOf("/gpio/1") != -1)
    val = 1;
  else {
```

```
    Serial.println("invalid request");
    client.stop();
    return;
  }

  // Set GPIO2 according to the request
  digitalWrite(2, val);

  client.flush();

  // Prepare the response
  String s = "HTTP/1.1 200 OK\r\nContent-Type:
text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\nGPIO is now ";
  s += (val)?"high":"low";
  s += "</html>\n";

  // Send the response to the client
  client.print(s);
  delay(1);
  Serial.println("Client disonnected");

  // The client will actually be disconnected
  // when the function returns and 'client' object is detroyed
}
```

Conclusion:

The GPIO of the ESP is used to successfully blink a Led using the well-known Arduino programming. In this way, by connecting the ESP to internet, many IoT based standalone projects can be developed.

References:

[1] http://www.whatimade.today/esp8266-easiest-way-to-program-so-far/

[2] http://iot-playground.com/blog/2-uncategorised/38-esp8266-and-arduino-ide-blink-example

Experiment 38: IoT based Smart Dustbin using NodeMCU

Aim:

To create an IoT based Smart Dustbin product using NodeMCU.

Objective:

To learn about I2C Protocol and how to interface IR Obstacle sensor and I2C OLED screen with NodeMCU and develop a standalone IoT based Smart Dustbin product which will send tweet alert when the dustbin needs to be emptied.

Components Required:

1. NodeMCU
2. I2C 0.96" 128X64 White OLED Display
3. IR obstacle sensor
4. USB Connector
5. Jumper Wires

Connections:

NodeMCU	IR Obstacle sensor
3.3v	VCC
GND	GND
D5	OUT

NodeMCU	OLED Display
3.3v	VCC
GND	GND
D1	SCL
D2	SDA

7. Perform the following connections as shown in the table.

Theory:

- The Inter-integrated Circuit (I2C) Protocol is a protocol intended to allow multiple "slave" digital integrated circuits ("chips") to communicate with one or more "master" chips. Like the Serial Peripheral Interface (SPI), it is only intended for short distance communications within a single device. Like Asynchronous Serial Interfaces (such as RS-232 or UARTs), it only requires two signal wires to exchange information.

- There are two types of Display available: The SPI Display and I2C Display. The SPI is faster but uses more wires for connections and thus using up more pins of the Arduino. The I2C Display uses only 2 wires for communication.

- NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

- The IR Obstacle sensor indicates when the threshold is reached and when its time to empty.

Procedure:

50. Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows us to display on the OLED screen.
51. Go to link provided in Reference [1] and Reference [2] and download the OLED libraries.
52. Next in the Arduino IDE, click on:
53. *Sketch >> Include Library >> Add .ZIP library.*
54. Browse and select the folder where you downloaded the libraries.
55. Next, open the Arduino IDE and go to files and click on the preference in the Arduino IDE.
    Files >> Preference >> Additional Boards Manager URLs: paste
    http://arduino.esp8266.com/stable/package_esp8266com_index.json >> Ok
56. Next click on
    Tools >> Board >> Boards Manager >> search for esp8266 >> Navigate to esp8266 by Esp8266 community and install the software for Arduino.
57. Go to Thingspeak and create an account. Link provided in Reference [6].
58. Create a new channel and fill in the details :
    Field1 : Dustbin Status
59. Get your private key from the channel and paste it in the code given below and also enter your own ssid of wifi and password.
60. Plug in your NodeMCU with the connections made as shown in the table above. Select the Board NodeMCU 1.0 (ESP-12E module) and upload the code given below.

Note: You have to press the reset button every time you want to upload code to the NodeMCU Board. The on board blue LED blinks once in order to indicate when you should hit the reset button.

61. Check in your channel if the values 1 and 0 are displayed according to the obstacle sensing. Open the Serial Monitor for debugging purposes.

62. Now, once the values are displayed, we need to send out tweet alerts. In order to do so follow the steps below:

63. Create a new Twitter account for yourself. Next in Thingspeak click on
APPS >> ThingTweet >> Link Twitter Account.

64. Once you've linked your Twitter account, Click on
APPS >> React >> New React

65. In the space 'then tweet', enter the following:
The dustbin needs to be emptied!

66. Whenever the value is equal to 1, you will get a tweet alert reminding you to empty the dustbin!

- *Debugging:*
  - *If you receive an error while compiling stating Height incorrect, perform the following task:*
    Got to Your PC >> Documents >> Arduino >> libraries >> Adafruit SSD1306 >> Adafruit ssd1306.h >> Edit with notepad.
    Uncomment the line: #define SSD1306_128_64
    And comment the line: #define SSD1306_128_32
    This will fix the height error bug.
  - *If you receive an error stating no slave found at the address, that may be because your OLED Display may have a different slave address. In order to find out, with the connections made to OLED display, upload the code given in Reference [4] and Check your slave address on Serial monitor. Change the address and output displayed on OLED Screen.*

Source Code:

```
/**
 * IoT based Smart Dustbin using NodeMCU.
 * Code developed by : Ms. Michelle D'Souza, IoT Intern, 2016.
 * Grateful to the whole open source community on internet.
*/
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <ESP8266WiFi.h>

#define OLED_RESET 3//4 for pro mini
Adafruit_SSD1306 display(OLED_RESET);


#if (SSD1306_LCDHEIGHT != 64)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif
int isObstaclePin = 2;  // This is our input pin
```

```
int isObstacle = HIGH;  // HIGH MEANS NO OBSTACLE

const char* ssid     = "xxxxxx";
const char* password = "xxxxxxxxxxxx";
const char* host = "184.106.153.149";//api.thingspeak.com
const char* privateKey = "XXXXXXXXXXXXXXX";
int distance;
String T = "Time to Empty!!";
String A = "Accepting Waste :)";
String stat;

void setup() {
 pinMode(isObstaclePin, INPUT);

 Serial.begin(9600);
 Serial.println("DHTxx test!");

 display.begin(SSD1306_SWITCHCAPVCC, 0x78 >> 1); // set display
 // display.display(); //Show the Adafruit splash screen!
 //delay(20);
 display.clearDisplay();
 display.setTextSize(2);
 display.setTextColor(WHITE);
 display.setCursor(0, 0);

 display.println("Smart");
 display.display();

 display.print("Dustbin");

 display.display();
 delay(1000);
 Serial.print("Connecting to ");
 Serial.println(ssid);

 display.clearDisplay();
 display.setCursor(0, 0);
 display.setTextSize(1);
 display.println("Connecting to WiFI...");
 display.println();
 display.display();

 WiFi.begin(ssid, password);

 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
 }
```

```
 display.println("Connected at IP: ");
 display.print(WiFi.localIP());
 display.display();
 delay(1000);

}
int i;
void loop() {
 display.clearDisplay();
 display.setTextSize(1);
 display.setTextColor(WHITE);
 display.setCursor(0, 0);

 isObstacle = digitalRead(isObstaclePin);
 if (isObstacle == HIGH)
 {
  Serial.println("OBSTACLE!!");
  display.println(T);
  display.println();
  display.display();
  stat = T;
  distance = 1;
  updateData();

  // digitalWrite(LED, HIGH);
 }
 else
 {
  Serial.println("clear");
  display.println(A);
  display.println();
  display.display();
  stat = A;
  distance = 0;
  updateData();

  // digitalWrite(LED, LOW);
 }

 display.clearDisplay();
 display.setCursor(0, 0);
 display.setTextSize(1);
 display.println(stat);
 display.println();
 display.println("Next update in >>");
 display.println();
 display.display();
 delay(500);
 for (i = 20 ; i > 0 ; i--) {
```

```
   display.print(i);
   display.print(" ");
   display.display();
   delay(1000);
  }
}

void updateData() {
 Serial.print("distance");

 Serial.println(distance);
 //GET http://api.thingspeak.com/update?key=XXXXXXXXXXXXXXX&field1=98&field2=29&field3=23
 Serial.print("connecting to ");
 Serial.println(host);
 display.setTextSize(1);
 display.setTextColor(WHITE);
 // display.setCursor(0, 10 );
 display.println("Uploading data");
 display.drawPixel(30, 30, WHITE);
 display.display();

 // Use WiFiClient class to create TCP connections
 WiFiClient client;
 const int httpPort = 80;
 if (!client.connect(host, httpPort)) {
  Serial.println("connection failed");
  return;
 }
 display.drawPixel(50, 30, WHITE);
 display.display();
 // We now create a URI for the request
 String url = "/update?key=";
 url += privateKey;
 url += "&field1=";
 url += distance;

 Serial.print("Requesting URL: ");
 Serial.println(url);

 // This will send the request to the server
 client.print(String("GET ") + url + " HTTP/1.1\r\n" +
         "Host: " + host + "\r\n" +
         "Connection: close\r\n\r\n");
 unsigned long timeout = millis();
 display.drawPixel(70, 30, WHITE);
 display.display();
 while (client.available() == 0) {
  if (millis() - timeout > 5000) {
   Serial.println(">>> Client Timeout !");
   client.stop();
```

```
    return;
   }
  }

  // Read all the lines of the reply from server and print them to Serial
  while (client.available()) {
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }
  display.println();
  display.println();
  display.println("Uploaded!");
  display.display();
  delay(1000);
  display.clearDisplay();
  Serial.println();
  Serial.println("closing connection");
}
```

Applications:

Many such modules combined together to develop a smart dustbin system for hospitals to dispose waste efficiently.

Conclusion:

IoT based smart Dustbin is developed and output viewed on OLED screen and alerts tweeted.

References:

[1] https://github.com/adafruit/Adafruit-GFX-Library

[2] https://github.com/adafruit/Adafruit_SSD1306

[3] https://twitter.com/

[4] http://playground.arduino.cc/Main/I2cScanner

[5] http://internetofthinking.blogspot.in/2015/12/getting-started-with-esp8266-nodemcu.html

[6] https://thingspeak.com/

Experiment 39: Display DHT11 sensor data on OLED screen using NodeMCU.

Aim:

To display DHT11 sensor data on OLED Screen using NodeMCU.

Objective:

To learn about I2C Protocol and how to interface DHT11 sensor and I2C OLED screen with NodeMCU and develop a stand-alone sensor display system.

Components Required:

1. NodeMCU
2. I2C 0.96" 128X64 White OLED Display
3. DHT11 Sensor Module
4. Jumper Wires
5. USB Connector

Connections:

NodeMCU	DHT11 Sensor
3.3v	VCC
GND	GND
D5	OUT

NodeMCU	OLED Display
3.3v	VCC
GND	GND
D1	SCL
D2	SDA

Perform the following connections as shown in the table.

Theory:

- The Inter-integrated Circuit (I2C) Protocol is a protocol intended to allow multiple "slave" digital integrated circuits ("chips") to communicate with one or more "master" chips. Like the Serial Peripheral Interface (SPI), it is only intended for short distance communications within a single device. Like Asynchronous Serial Interfaces (such as RS-232 or UARTs), it only requires two signal wires to exchange information.

- There are two types of Display available: The SPI Display and I2C Display. The SPI is faster but uses more wires for connections and thus using up more pins of the Arduino. The I2C Display uses only 2 wires for communication.

- NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

Procedure:

67. Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows us to display on the OLED screen.
68. Go to link provided in Reference [1] and Reference [2] and download the OLED libraries.
69. Next in the Arduino IDE, click on:
70. *Sketch >> Include Library >> Add .ZIP library.*
71. Browse and select the folder where you downloaded the libraries.
72. Go to link provided in Reference [3] and download the DHT11 library. Follow the instructions in their notes section if you're facing any problems.
73. Next, open the Arduino IDE and go to files and click on the preference in the Arduino IDE. *Files >> Preference >> Additional Boards Manager URLs: paste* *http://arduino.esp8266.com/stable/package_esp8266com_index.json* *>> Ok*
74. Next click on
    Tools >> Board >> Boards Manager >> search for esp8266 >> Navigate to esp8266 by Esp8266 community and install the software for Arduino.
75. Plug in your NodeMCU with the connections made as shown in the table above. Select the Board NodeMCU 1.0 (ESP-12E module) and upload the code given below.

Note: You have to press the reset button every time you want to upload code to the NodeMCU Board. The on board blue LED blinks once in order to indicate when you should hit the reset button.

76. Open the Serial Monitor for debugging purposes.
77. If the OLED Display is not working then follow the steps:

- *File >> Examples >> Adafruit SSD1306 >> ssd1306_128x64_i2c.*
- *Debugging:*
  - *If you receive an error while compiling stating Height incorrect, perform the following task:*

    Got to Your PC >> Documents >> Arduino >> libraries >> Adafruit SSD1306 >> Adafruit ssd1306.h >> Edit with notepad.

    Uncomment the line: #define SSD1306_128_64

    And comment the line: #define SSD1306_128_32

    This will fix the height error bug.
  - *If you receive an error stating no slave found at the address, that may be because your OLED Display may have a different slave address. In order to find out, with the connections made to OLED display, upload the code given in Reference [4] and Check your slave address on Serial monitor.*
  - *Change the address and output displayed on OLED Screen.*

Source Code:

```
/**
   Display DHT11 sensor value on OLED using NodeMCU.
   Code modified by : Ms. Michelle D'Souza, IoT Intern, 2016.
   Grateful to the whole open source community on internet.
*/

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
//#include <SPI.h>
//#include <Wire.h>
#include "DHT.h"


#define OLED_RESET 3//4 for pro mini
Adafruit_SSD1306 display(OLED_RESET);

#define DHTPIN 14     // what digital pin we're connected to
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);

#if (SSD1306_LCDHEIGHT != 64)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif

void setup()   {
  Serial.begin(9600);
  // by default, we'll generate the high voltage from the 3.3v line
internally! (neat!)
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C
addr 0x3D (for the 128x64)
  // init done
```

```
  dht.begin();
  // Clear the buffer containing adafruits splashscreen.
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("DHT11 Sensor Values");
  display.display();
  delay(2000);

}


void loop() {
  // Get the values from sensor
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.print("Temperature : ");
  display.print(t);
  display.println("*C");
  display.println();

  display.print("Humidity : ");
  display.print(h);
  display.println("%");
  display.display();
  delay(2000);
  display.clearDisplay();
  display.print("Reading value... ");
  display.display();
}
```


Applications:

Sensor data display system using NodeMCU.

Conclusion:

Sensor data of DHT11 is displayed on OLED Display using NodeMCU.

References:

[1] https://github.com/adafruit/Adafruit-GFX-Library

[2] https://github.com/adafruit/Adafruit_SSD1306

[3] https://github.com/adafruit/DHT-sensor-library

[4] http://playground.arduino.cc/Main/I2cScanner

[5] http://internetofthinking.blogspot.in/2015/12/getting-started-with-esp8266-nodemcu.html

[6] https://www.youtube.com/watch?v=RpK7-Ljnpho

[7] http://www.14core.com/wiring-dht11-dht22-with-oled-screen-on-esp8266-12e/

Experiment 40: Send data from various sensors to ThingSpeak using NodeMCU

Aim:

To send data from various sensors to ThingSpeak channel using NodeMCU.

Objective:

To learn about I2C Protocol and how to interface DHT11 sensor, Ultrasonic sensor and I2C OLED screen with NodeMCU and develop a standalone IoT system by sending data over the internet.

Components Required:

1. NodeMCU
2. I2C 0.96" 128X64 White OLED Display
3. DHT11 Sensor Module
4. Ultrasonic Sensor
5. USB Connector
6. Jumper Wires

Connections:

NodeMCU	DHT11 Sensor
3.3v	VCC
GND	GND
D5	OUT

NodeMCU	OLED Display
3.3v	VCC
GND	GND
D1	SCL
D2	SDA

NodeMCU	Ultrasonic Sensor
5v (P.S)	VCC
GND	GND
D7	Echo

D6	Trig

Perform the following connections as shown in the table.
Note: Power up the ultrasonic sensor with an external Power Supply of 5v and don't forget to short the grounds!

Theory:

- The Inter-integrated Circuit (I2C) Protocol is a protocol intended to allow multiple "slave" digital integrated circuits ("chips") to communicate with one or more "master" chips. Like the Serial Peripheral Interface (SPI), it is only intended for short distance communications within a single device. Like Asynchronous Serial Interfaces (such as RS-232 or UARTs), it only requires two signal wires to exchange information.

- There are two types of Display available: The SPI Display and I2C Display. The SPI is faster but uses more wires for connections and thus using up more pins of the Arduino. The I2C Display uses only 2 wires for communication.

- NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

Procedure:

78. Once you have your Arduino IDE installed and updated to the latest version, we will use an additional library which allows us to display on the OLED screen.
79. Go to link provided in Reference [1] and Reference [2] and download the OLED libraries.
80. Next in the Arduino IDE, click on:
81. *Sketch >> Include Library >> Add .ZIP library.*
82. Browse and select the folder where you downloaded the libraries.
83. Go to link provided in Reference [3] and download the DHT11 library. Follow the instructions in their notes section if your facing any problems.
84. Next, open the Arduino IDE and go to files and click on the preference in the Arduino IDE.
    Files >> Preference >> Additional Boards Manager URLs: paste
    http://arduino.esp8266.com/stable/package_esp8266com_index.json >> Ok
85. Next click on

    Tools >> Board >> Boards Manager >> search for esp8266 >> Navigate to esp8266 by Esp8266 community and install the software for Arduino.

86. Go to ThingSpeak and create an account. Link provided in Reference [6].
87. Create a new channel and fill in the details :
    Field1: Distance
    Field2: Temperature
    Field3: Humidity

88. Get your private key from the channel and paste it in the code given below and also enter your own ssid of wifi and password.
89. Plug in your NodeMCU with the connections made as shown in the table above. Select the Board NodeMCU 1.0 (ESP-12E module) and upload the code given below.

Note: You have to press the reset button every time you want to upload code to the NodeMCU Board. The on board blue LED blinks once in order to indicate when you should hit the reset button.

90. Open the Serial Monitor for debugging purposes.
91. If the OLED Display is not working then follow the steps:

- *File >> Examples >> Adafruit SSD1306 >> ssd1306_128x64_i2c.*
- *Debugging:*
    - *If you receive an error while compiling stating Height incorrect, perform the following task:*
      Got to Your PC >> Documents >> Arduino >> libraries >> Adafruit SSD1306 >> Adafruit ssd1306.h >> Edit with notepad.
      Uncomment the line: #define SSD1306_128_64
      And comment the line: #define SSD1306_128_32
      This will fix the height error bug.
    - *If you receive an error stating no slave found at the address, that may be because your OLED Display may have a different slave address. In order to find out, with the connections made to OLED display, upload the code given in Reference [4] and Check your slave address on Serial monitor. Change the address and output displayed on OLED Screen.*

Source Code:

```
/**
 * Send sensor data to ThingSpeak using NodeMCU.
 * Code developed by : Ms. Michelle D'Souza, IoT Intern, 2016.
 * Grateful to the whole open source community on internet.
 * http://www.14core.com/wiring-dht11-dht22-with-oled-screen-on-esp8266-12e/
 * http://www.14core.com/wiring-esp8266-nodemcu-with-hcsr04-ultrasonic-sensor/
*/

//#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "ESP8266WiFi.h"
```

```
#include "DHT.h"
#define OLED_RESET 3

Adafruit_SSD1306 display(OLED_RESET);
#define DHTPIN 14     // Connected to Pin D5 in NodeMCU
#define TRIGGER 12    // Connected to Pin D6 in NodeMCU
#define ECHO    13    // Connected to Pin D7 in NodeMCU
// Uncomment whatever type you're using!
#define DHTTYPE DHT11   // DHT 11
//#define DHTTYPE DHT22   // DHT 22 (AM2302)
//#define DHTTYPE DHT21   // DHT 21 (AM2301)

DHT dht(DHTPIN, DHTTYPE, 15);

long duration, distance;
float h, t;
const char* ssid     = "xxxx";//Enter your own
const char* password = "xxxxx";//Enter your own
const char* host = "184.106.153.149";//api.thingspeak.com
const char* privateKey = "XXXXXXXXXXXXXXXXXX";//Enter your own private key
byte img [] = {
  0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0,
  0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf0, 0x0,
0x0,
  0x0, 0x0, 0x3f, 0xfc, 0x0, 0x0, 0x0, 0x0, 0xff, 0xff, 0x0, 0x0, 0x0, 0x1,
0xfc,
  0x3f, 0x80, 0x0, 0x0, 0x3, 0xe0, 0x7, 0xc0, 0x0, 0x0, 0x7, 0x80, 0x1,
0xe0, 0x0,
  0x0, 0xff, 0x0, 0x0, 0xff, 0x0, 0x3, 0xfe, 0x0, 0x0, 0x7f, 0xc0, 0x7,
0xe2, 0x0,
  0x0, 0x4f, 0xe0, 0x7, 0x80, 0x0, 0x0, 0x1, 0xe0, 0xf, 0x0, 0x0, 0x0, 0x0,
0xf0,
  0x1e, 0x0, 0x0, 0x0, 0x0, 0x70, 0xe, 0x0, 0x0, 0x0, 0x0, 0x78, 0xc, 0x0,
0x0,
  0x0, 0x0, 0x30, 0xc, 0x0, 0x0, 0x0, 0x0, 0x30, 0xc, 0x0, 0x0, 0x0, 0x0,
0x30,
  0xe, 0x0, 0x0, 0x0, 0x0, 0x70, 0x1e, 0x0, 0x0, 0x0, 0x0, 0x70, 0xf, 0x0,
0x0,
  0x0, 0x0, 0xf0, 0x7, 0x80, 0x0, 0x0, 0x1, 0xe0, 0x7, 0xe3, 0x80, 0x1,
0xc7, 0xe0,
  0x3, 0xff, 0xc0, 0x3, 0xff, 0xc0, 0x0, 0xff, 0xf0, 0xf, 0xff, 0x0, 0x0,
0x20, 0xfc,
  0x3f, 0x4, 0x0, 0x0, 0x0, 0x3f, 0xfc, 0x0, 0x0, 0x0, 0x0, 0xf, 0xf0, 0x0,
0x0,
  0x0, 0x4, 0x0, 0x0, 0x0, 0x0, 0x0, 0x4, 0x0, 0x0, 0x0, 0x0, 0x0, 0xe,
0x0,
  0x0, 0x0, 0x0, 0x0, 0xe, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1f, 0x0, 0x0, 0x0,
0x0,
  0x0, 0x1f, 0x0, 0x0, 0x20, 0x0, 0x0, 0x1f, 0x0, 0x0, 0x20, 0x0, 0x0, 0xe,
0x0,
  0x0, 0x70, 0x0, 0x0, 0x0, 0x1, 0x80, 0x70, 0x0, 0x0, 0x0, 0x1, 0x80,
0xf8, 0x0,
  0x0, 0x0, 0x3, 0xc0, 0xf8, 0x0, 0x0, 0x0, 0x3, 0xc0, 0xf8, 0x0, 0x0, 0x0,
0x3,
  0xc0, 0x70, 0x0, 0x0, 0x0, 0x3, 0xc0, 0x0, 0x0, 0x0, 0x0, 0x3, 0xc0, 0x0,
0x0,
  0x0, 0x0, 0x1, 0x80, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0,
```

```
  0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0,

};

void setup() {
  pinMode(TRIGGER, OUTPUT);
  pinMode(ECHO, INPUT);

  Serial.begin(9600);
  Serial.println("DHTxx test!");

  dht.begin();
  display.begin(SSD1306_SWITCHCAPVCC, 0x78 >> 1); // set display
  // display.display(); //Show the Adafruit splash screen!
  //delay(20);
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    display.clearDisplay();//Clear the Adafruit splash screen!
    display.setTextSize(2);
    display.setTextColor(WHITE);
    display.setCursor(20, 30);
    display.println("Loading");
    display.drawPixel(40, 50, WHITE);
    display.display();
    display.drawPixel(50, 50, WHITE);
    display.display();
    display.drawPixel(60, 50, WHITE);
    display.display();
    delay(500);
    display.clearDisplay();
  }

  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void showTemp(float temp, float hud) {
  display.drawBitmap(0, 5,  img, 48, 50, 1);
  display.setTextSize(1);
  display.setCursor(3, 0);
  display.println("Temperature/Humidity");
  display.display();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(52, 10);
  display.print(temp);
  display.println("C");
  display.display();
  display.setCursor(52, 30);
  display.print(hud);
  display.println("%");
  display.display();
```

```
  display.setTextSize(1);
  display.setCursor(0, 50);
  display.println("Awesomeness Guaranteed!");
  display.display();
  delay(1000);
  display.clearDisplay();
}
void loop() {

  delay(1000);   // Wait a few seconds between measurements.
  h = dht.readHumidity();
  t = dht.readTemperature();   // Read temperature as Celsius

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    h = t = 0;
    // return;
  }
  Serial.print("Humidity: ");    // show in serial monitor
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");    // show in serial monitor
  Serial.print(t);
  Serial.print(" *C \n");
  showTemp(t, h);  // show temp

  digitalWrite(TRIGGER, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER, LOW);
  duration = pulseIn(ECHO, HIGH);
  distance = (duration / 2) / 29.1;
  Serial.print(distance);
  Serial.println("Centimeter:");
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("Distance:"); //display.println(0xDEADBEEF, HEX);
  display.display();
  display.setTextSize(2);
  display.print(distance);
  display.print("cm");
  display.display();
  delay(2000);
  display.clearDisplay();
  updateData();

}

void updateData() {
  //GET
http://api.thingspeak.com/update?=XXXXXXXXXXXXXXXX&field1=98&field2=29&fiel
d3=23
  Serial.print("connecting to ");
  Serial.println(host);
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
```

```
  display.println("Uploading data");
  display.drawPixel(30, 40, WHITE);
  display.display();

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }
  display.drawPixel(50, 40, WHITE);
  display.display();
  // We now create a URI for the request
  String url = "/update?key=";
  url += privateKey;
  url += "&field1=";
  url += distance;
  url += "&field2=";
  url += t;
  url += "&field3=";
  url += h;

  Serial.print("Requesting URL: ");
  Serial.println(url);

  // This will send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  unsigned long timeout = millis();
  display.drawPixel(70, 40, WHITE);
  display.display();
  while (client.available() == 0) {
    if (millis() - timeout > 5000) {
      Serial.println(">>> Client Timeout !");
      client.stop();
      return;
    }
  }

  // Read all the lines of the reply from server and print them to Serial
  while (client.available()) {
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }
  display.clearDisplay();
  display.println("Uploaded!");
  display.display();
  delay(1000);
  display.clearDisplay();
  Serial.println();
  Serial.println("closing connection");
}
```

Applications:

Sensor data display system using NodeMCU.

Conclusion:

Sensor data of DHT11 is displayed on OLED Display using NodeMCU.

References:

[1] https://github.com/adafruit/Adafruit-GFX-Library

[2] https://github.com/adafruit/Adafruit_SSD1306

[3] http://playground.arduino.cc/Main/DHT11Lib

[4] http://playground.arduino.cc/Main/I2cScanner

[5] http://internetofthinking.blogspot.in/2015/12/getting-started-with-esp8266-nodemcu.html

[6] https://thingspeak.com/

[7] http://www.14core.com/wiring-dht11-dht22-with-oled-screen-on-esp8266-12e/

[8] http://www.14core.com/wiring-esp8266-nodemcu-with-hcsr04-ultrasonic-sensor/

Experiment 41: To send an SMS if object detected (Raspberry Pi)

Aim:

To send an SMS if distance is less than a threshold value using distance sensor and RPi.

Objective:

To learn how to use various cross protocol communication techniques to achieve the aim of the experiment.

Components Required:

1. Raspberry Pi and power supply adapter.
2. Screen to view the RPi desktop:
   - HDMI to VGA cable and Computer Monitor or
   - Touchscreen Display for RPi or
   - View the RPi screen remotely using VNC viewer .
3. HC-SR04 Ultrasonic Sensor.
4. Jumper wires.
5. Resistors.

Connections:

1. Connect the power supply micro USB cable and the HDMI port of the HDMI to VGA cable to the RPi.
2. Connect the VGA port of the HDMI to VGA cable to any computers monitor's VGA port.
3. Thats all, now switch on the power supply.
4. If you want to view the RPi screen remotely using VNC server, follow the steps given in Reference [2].

Note: Do not use any mobile charger adapter to power on the RPi. Use 5V-2A for RPi 2 and 5V-2.5A for RPi 3.

5. To interface the distance sensor, follow the connections shown in the diagram below.

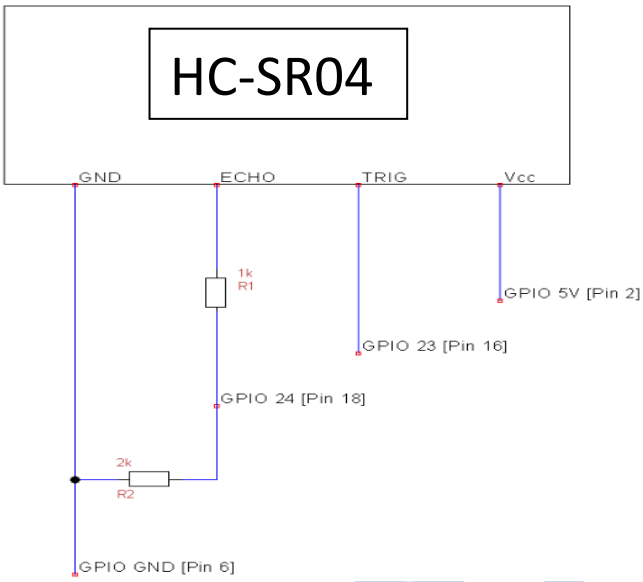


Fig:HC-SR04 connections with RPi.

Courtesy: https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

Procedure:

Once you've set up your connections with the Pi, complete these steps before uploading the code on the Pi.

1. You will first need to set up an account on : *https://www.twilio.com/*

   (You will be asked to fill in your details and verify your phone number)

2. After you've successfully created an account, you will find your ACCOUNT SID and AUTH TOKEN values which you will require to copy paste in the code given below.

3. Next you will have to use a free Twilio number to send text messages. So, go to : *https://www.twilio.com/console/phone-numbers/incoming* and get your first free Twilio number.

4. Once you know your free Twilio number, Copy paste it in the code in the same format and also enter your registered phone number in the code given below.

5. Create a python file in your RPi and copy paste the code with your details edited. Your just one last step away before running the code and receiving an SMS.

6. In your RPi terminal, copy paste the following line to install a package :

   Command: sudo pip install twilio.

   (If you are facing any problems during installation, check this page for ddebugging purposes: *https://www.twilio.com/docs/libraries/python*)

7. All set and ready to go. Save and run your python script. Observe the distance on the python shell and watch the Pi shoot an SMS if an obstacle is detected less than 3 cm.

Source Code:

```
#This code will send an SMS if distance is less then threshold value
detected by distance sensor.
import RPi.GPIO as GPIO
import time
from twilio.rest import TwilioRestClient

GPIO.setmode(GPIO.BCM)

TRIG = 23
ECHO = 24

# put your own credentials here
ACCOUNT_SID = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
AUTH_TOKEN = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)

GPIO.output(TRIG, False)
print "Distance Measurement In Progress"

print "Waiting For Sensor To Settle"
time.sleep(2)
while True:
  GPIO.output(TRIG, True)
  time.sleep(0.00001)
  GPIO.output(TRIG, False)

  while GPIO.input(ECHO)==0:
    pulse_start = time.time()

  while GPIO.input(ECHO)==1:
    pulse_end = time.time()

  pulse_duration = pulse_end - pulse_start

  distance = pulse_duration * 17150


  distance = round(distance, 2)

  print "Distance:",distance,"cm"
  time.sleep(1)

  if distance<3:
      print"alert!!sending sms"
      client = TwilioRestClient(ACCOUNT_SID, AUTH_TOKEN)
      client.messages.create(
          to="+919876543210",#Your registered number
          from_="+13216549870",#your twilio number
          body="ultrasonic sensor :distance less then 3cm!!",
```

```
)

GPIO.cleanup()
```

Applications:

- Easy Graphical User Interface to control appliances of any organization or industry .

Conclusion:

With the use of Raspberry Pi, appliances can be controlled locally or remotely using the internet using a creative graphical interface..

References:

[1] https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

[2] https://www.raspberrypi.org/documentation/remote-access/vnc/README.md

[3] https://www.twilio.com/

Experiment 42: IoT based image tweeted when camera detects motion using Raspberry Pi.

Aim:

To tweet an image when the camera detects motion from Raspberry Pi.

Objective:

To learn how to use Twitter App to automatically tweet images from Raspberry Pi.

Components Required:

1. Raspberry Pi and power supply adapter.
2. Screen to view the RPi desktop:
   - HDMI to VGA cable and Computer Monitor or
   - Touchscreen Display for RPi or
   - View the RPi screen remotely using VNC viewer.
3. Raspberry Pi Camera

Connections:

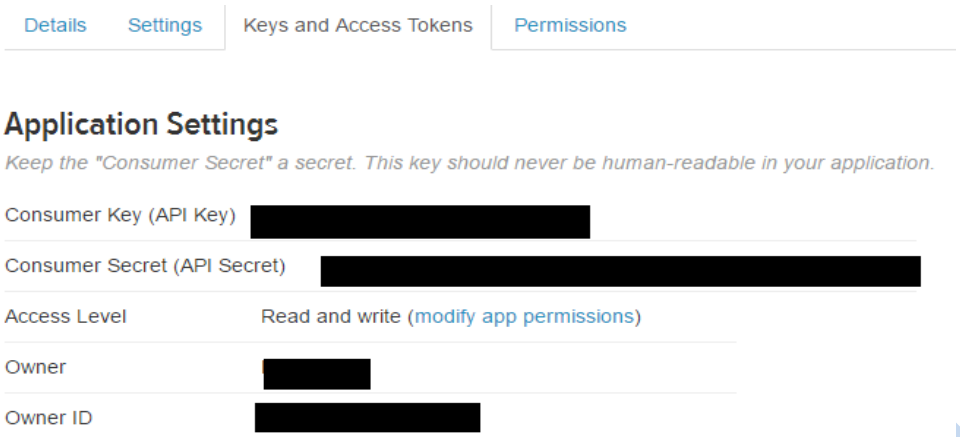
1. Connect the power supply micro USB cable and the HDMI port of the HDMI to VGA cable to the RPi.
2. Connect the VGA port of the HDMI to VGA cable to any computers monitor's VGA port.
3. That's all, now switch on the power supply.
4. If you want to view the RPi screen remotely using VNC server, follow the steps given in Reference [2].

Note: Do not use any mobile charger adapter to power on the RPi. Use 5V-2A for RPi 2 and 5V-2.5A for RPi 3.

Procedure:

1. Begin by creating a new Twitter account and then go to link provided in reference [1] and click create a new app.

2. Fill in all the necessary details and proceed. Next click on *Keys And Access Tokens >> Create new Access key*



3. Note down your Consumer Key (API Key), Consumer Secret (API Secret), Access Token, Access Token Secret

4. Now,once your RPi is up and ready, you will need to download and install a Twitter App that will help us tweet images. To do so, go to the RPi terminal and type the following:

   Command: sudo pip install tweepy

5. Copy paste the contents of the code given below and change the above values according to yours where required.

6. Now next enable the camera on the raspberry pi and after switching off the Pi, connect the camera and lock it carefully. Please go to link provided in reference[3] for getting started with the picamera.

7. Make sure you have internet connection on the Pi and run the python script.

Source Code:

```
#!/usr/bin/env python2.7
# Image tweeted when camera detects motion using Raspberry Pi.
# -*- coding: utf-8 -*-
# Modified by: Ms.Michelle D'Souza, IoT Intern, 2016.
# Grateful to the whole open source community on internet.
# by Alex Eames http://raspi.tv/?p=5918

import tweepy
from subprocess import call
from datetime import datetime
import RPi.GPIO as GPIO
import time
```

```
from picamera import PiCamera

# Consumer keys and access tokens, used for OAuth
access_token = 'Your Acess Token'
access_token_secret = 'Your secret acess token'

consumer_key = 'Your Consumer key'
consumer_secret = 'Your consumer secret'

# OAuth process, using the keys and tokens
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

# Creation of the actual interface, using authentication
api = tweepy.API(auth)
```

```
GPIO.setmode(GPIO.BOARD)
PIR_PIN = 7
GPIO.setup(PIR_PIN, GPIO.IN)
camera = PiCamera()

i = datetime.now()            #take time and date for filename
now = i.strftime('%Y%m%d-%H%M%S')

print ("Twitter Image Test CTRL+C to exit")
time.sleep(1)
print ("Ready?")

try:
   while True:
      if GPIO.input(PIR_PIN):
         print ("Motion Detected!")
         time.sleep(1)
         #camera.start_preview()
         camera.annotate_text = "Image captured from Pi Camera!"
         time.sleep(5)
         filename = datetime.now().strftime("%Y-%m-%d_%H.%M.%S.jpg")
         camera.capture(filename)
         #camera.stop_preview()
         photo_path = '/home/pi/' + filename
         status = 'Photo auto-tweet from RPi when motion detected: ' + i.strftime('%Y/%m/%d
%H:%M:%S')
         api.update_with_media(photo_path, status=status)
         print ("Image Captured!")
      else:
         print ("all clear")

except KeyboardInterrupt:
      print ("Quit")
      GPIO.cleanup()
```

Applications:

- A Tweet Alert system can be created for any application where text or images can be tweeted.

Conclusion:

Images are automatically tweeted from Pi when motion detected.

References:

[1] https://apps.twitter.com/

[2] https://www.raspberrypi.org/documentation/remote-access/vnc/README.md

[3] https://www.raspberrypi.org/learning/getting-started-with-picamera/worksheet/

[4] https://www.raspberrypi.org/learning/parent-detector/worksheet/

[5] http://raspi.tv/2013/how-to-create-a-twitter-app-on-the-raspberry-pi-with-python-tweepy-part-1#app

Experiment 43 : Graphical User Interface to control appliances using Raspberry Pi

Aim:

To create a graphical user interface in order to control appliances.

Objective:

To learn how to integrate code of graphical user interface and GPIO ports of RPi using python script.

Components Required:

1. Raspberry Pi and power supply adapter.
2. Screen to view the RPi desktop:
   - HDMI to VGA cable and Computer Monitor or
   - Touchscreen Display for RPi or
   - View the RPi screen remotely using VNC viewer .

Connections:

1.Connect the power supply micro USB cable and the HDMI port of the HDMI to VGA cable to the RPi.
2.Connect the VGA port of the HDMI to VGA cable to any computers monitor's VGA port.
3.Thats all, now switch on the power supply.
4.If you want to view the RPi screen remotely using VNC server, follow the steps given in Reference [2].
Note: Do not use any mobile charger adapter to power on the RPi. Use 5V-2A for RPi 2 and 5V-2.5A for RPi 3.

Procedure:

Python provides various options for developing graphical user interface. In this experiment we will be using Tkinter.

Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin object-oriented layer on top of Tcl/Tk.

Simply switch on your raspberry pi and connect 2 leds to GPIO pin 38 and GPIO pin 40.

Open python shell. Create a new file and copy paste the code given below.

Click on the buttons on the graphical user interface screen and watch the leds glow!

Check reference [3] for further developments.

Source Code:

```
from Tkinter import *
import tkFont
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(40, GPIO.OUT)
GPIO.output(40, GPIO.LOW)
GPIO.setup(38, GPIO.OUT)
GPIO.output(38, GPIO.LOW)
win = Tk()

myFont = tkFont.Font(family = 'Helvetica', size = 36, weight = 'bold')

def ledON1():
    print("button1 pressed")
    if GPIO.input(40) :
        GPIO.output(40,GPIO.LOW)
        ledButton["text"] = "Button1 ON"
    else:
        GPIO.output(40,GPIO.HIGH)
                ledButton["text"] = "Button1 OFF"

def ledON2():
    print("button2 pressed")
    if GPIO.input(38) :
        GPIO.output(38,GPIO.LOW)
        ledButton["text"] = "Button2 ON"
    else:
        GPIO.output(38,GPIO.HIGH)
                ledButton["text"] = "Button2 OFF"


def exitProgram():
    print("Exit Button pressed")
        GPIO.cleanup()
    win.quit()


win.title("Switch ON/OFF")
win.geometry('800x480')

exitButton  = Button(win, text = "Exit", font = myFont, command =
exitProgram, height =2  , width = 6)
exitButton.pack(side = BOTTOM)

Button1 = Button(win, text = "LED ON", font = myFont, command = ledON1,
height = 2, width =8 )
Button1.pack()

Button2 = Button(win, text = "LED ON", font = myFont, command = ledON2,
height = 2, width =8 )
Button2.pack(side = RIGHT)
```

```
mainloop()

#http://educ8s.tv/raspberry-pi-gui-tutorial-create-your-own-gui-graphical-user-interface-with-tkinter-and-python/
```

Applications:

- Easy Graphical User Interface to control appliances of any organization or industry .

Conclusion:

With the use of Raspberry Pi, appliances can be controlled locally or remotely using the internet using a creative graphical interface..

References:

[1] http://educ8s.tv/raspberry-pi-gui-tutorial-create-your-own-gui-graphical-user-interface-with-tkinter-and-python/

[2] https://www.raspberrypi.org/documentation/remote-access/vnc/README.md

[3] http://www.tutorialspoint.com/python/python_gui_programming.htm

Experiment 44: Raspberry Pi GPIO Control via Webpage on LAN

Aim:

To create a webpage in order to control appliances connected to GPIO pins of Raspberry Pi via LAN.

Objective:

To learn how to control GPIO ports of RPi using html and php script and how to install apache server.

Components Required:

1. Raspberry Pi and power supply adapter.
2. Screen to view the RPi desktop:
   - HDMI to VGA cable and Computer Monitor or
   - Touchscreen Display for RPi or
   - View the RPi screen remotely using VNC viewer.

Connections:

1. Connect the power supply micro USB cable and the HDMI port of the HDMI to VGA cable to the RPi.
2. Connect the VGA port of the HDMI to VGA cable to any computers monitor's VGA port.
3. That's all; now switch on the power supply.
4. If you want to view the RPi screen remotely using VNC server, follow the steps given in Reference [2].

Note: Do not use any mobile charger adapter to power on the RPi. Use 5V-2A for RPi 2 and 5V-2.5A for RPi 3.

Procedure:

- In order to host any webpage we need a server. Apache is an open source and most widely used server.

  Note: Commands to be types in terminal are shown in red.

- Go to the RPis terminal and type the following:

sudo apt-get install apache2 –y

- Once the program is installed, we need to install php which allows server side actions to operate.

  sudo apt-get install php5 libapache2-mod-php5 –y

- After php is installed, we need to change ownership of the folder and all the files in it.

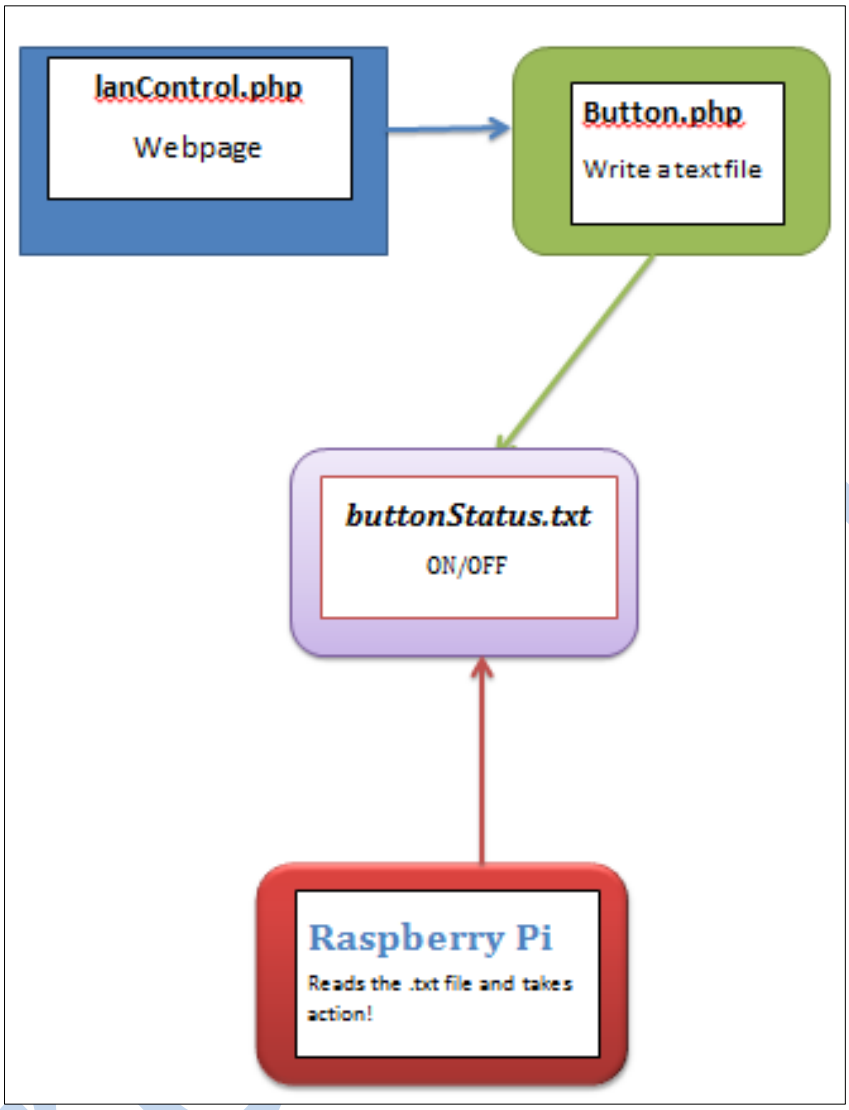
  sudo chown pi : pi /var/www/html

- Next type, to navigate into the html folder.

  sudo /var/www/html

- Save the *lanControl.php* file and *Button.php* file in this html folder.
- Also create a new buttonStatus.txt file in this html folder.
- Got to the browser and type *127.0.0.1/index.php*
- This should show you a webpage with 2 buttons to on/off the GPIO pin.
- Now, save the python script given below and connect an led and a resistor to GPIO pin 40.
- Now run the python code.
- Now click on the ON button in the browser and the led will glow.
- Remember always run the python code else the led will not glow.

- The following will help you understand the link between the programs:



Source Code:

```
<!--
Php code : lanControl.php
RPi Gpio Control on LAN
Code developed by : Ms. Michelle D'Souza, IoT Intern, 2016.
Grateful to the whole open source community on internet.
-->

<!DOCTYPE html>
<html>
    <head>
        <style>
        body {background-color:#fad8d8;font-family:'Lato',sans-
serif;color:#1c424d;text-align:center;}
        h1 {font-size:50px;}
        p {font-size:25px;}
        button{font-size:30px;}
        </style>
```

```
    </head>
    <body >
        <h1>Raspberry Pi - IoT</h1>
        <p>Switch ON/OFF your devices</p>
        <div class="container" >
                <form action="button.php" method="POST" role="form"
class="form-horizontal">
                    <button type="submit" name="on" class="btn btn-
default">Light1 on</button>
                    <button type="submit" name="off" class="btn btn-
default">Light1 off</button>
                    <br />
                    <br />
                </form>
        </div>

        <?php
        $myfile=fopen("ftpfile.txt","r") or die("unable to open file");
        echo fread($myfile,filesize("ftpfile.txt"));
        fclose($myfile);
        ?>*C

    </body>
</html>
```

```
<!--
Php code : Button.php
RPi Gpio Control on LAN
Code developed by : Ms. Michelle D'Souza, IoT Intern, 2016.
Grateful to the whole open source community on internet.
-->

<?php
 $file = "buttonStatus.txt";
 $handle = fopen($file,'w+');
if (isset($_POST['on'])) {
     $onstring = "ON";
     fwrite($handle,$onstring);
     fclose($handle);
     print "
     <!DOCTYPE html>
     <html>
     <head>
     <style>
     body{background-color:#fad8d8;font-family:'Lato',sans-
serif;color:#1c424d;text-align:center;}
     h1 {font-size:50px;}
     p {font-size:25px;}
     </style>
     </head>
     <body>
<h1>Raspberry Pi - IoT</h1> <p>The Light has been switched<strong>
ON</strong></p> </body> </html> ";
}
else if(isset($_POST['off'])) {
     $offstring = "OFF";
     fwrite($handle, $offstring);
```

```
    fclose($handle);
    print "
     <!DOCTYPE html>
     <html>
     <head>
     <style>
      body {background-color:#fad8d8;font-family:'Lato',sans-
serif;color:#1c424d;text-align:center;}
      h1 {font-size:50px;}
      p {font-size:25px;}
      </style>
      </head>
      <body>
<h1>Raspberry Pi - IoT</h1> <p>The Light has been switched<strong>
OFF</strong></p> </body> </html> ";
}
?>
```

```
#Python code
#RPi Gpio Control on LAN
#Code developed by : Ms. Michelle D'Souza, IoT Intern, 2016.
#Grateful to the whole open source community on internet.
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(40,GPIO.OUT)

while(1):
    try:
        f1=open("/var/www/html/buttonStatus.txt","r")
        status=f1.read()
        print status
        f1.close()

        if status=='ON':
            GPIO.output(40,False)
            print "ONing"
        elif status=='OFF':
            GPIO.output(40,True)
            print "Offing"

    except KeyboardInterrupt:
        GPIO.cleanup()
```

Applications:

- Easy Webpage Interface to control appliances of any organization or industry, locally or remotely using the internet if public IP address is available.
- Home Automation System.

Conclusion:

With the use of Raspberry Pi, appliances are controlled via webpage on LAN..

References:

[1] https://diyhacking.com/raspberry-pi-home-automation/

[2] https://www.raspberrypi.org/documentation/remote-access/vnc/README.md

Experiment 45: Get Door Status via the Internet using Raspberry Pi

Aim:

To check your door status on the internet (intialstate.com) remotely using Raspberry Pi 3.

Objective:

To learn to use the on-board WiFi module of the Raspberry Pi 3 to send data to the internet.

Components Required:

1. Raspberry PI 2 & SD Card with Raspbian Operating System
2. USB Power Supply
3. USB Cable
4. Breadboard & Jumper Wires
5. Door Sensor (Normally ON/OFF)

Connections: The Door Sensor has only 2 wires:

- One wire goes to the GPIO 23 (pin 16) of the Raspi.
- The other wire goes to the GND of the Raspi.

Procedure:

The first thing we need to install is the GPIO library. This library allows us to use the pins on the Pi and detect things like the magnetic switch.
First, make sure your Pi's software is up-to-date:

```
sudo apt-get update
```

Next, install a development kit that GPIO needs:

```
sudo apt-get install python-dev
```

Finally, install the library itself:

```
sudo apt-get install python-rpi.gpio
```

If you want to check that the switch is working before you try to stream anything, just replace **logger.log("Door","Open")** or **logger.log("Door","Close")** with **print("Door Open")** or **print("Door Close")** to see the output you get in the Pi's command line.

Source Code:

```
import time # so we can use "sleep" to wait between actions
import RPi.GPIO as io # import the GPIO library we just installed but call
it "io"
from ISStreamer.Streamer import Streamer # import the IS Streamer we just
installed but call it "Streamer"

## name the bucket and individual access_key
## the bucket_key will send all of our messages to the same place
## the access_key tells Initial State to send the messages to you
logger=Streamer(bucket_name="Locker
Protector",bucket_key="locker_protector",access_key="Your_Access_Key_Here")

## set GPIO mode to BCM
## this takes GPIO number instead of pin number
io.setmode(io.BCM)

## enter the number of whatever GPIO pin you're using
door_pin = 23

## use the built-in pull-up resistor
io.setup(door_pin, io.IN, pull_up_down=io.PUD_UP)  # activate input with
PullUp

## initialize door
door=0

## this loop will execute the if statement that is true
while True:
    ## if the switch is open
    if io.input(door_pin):
        logger.log("Door","Open") # stream a message saying "Open"
        logger.flush() # send the message immediately
        door=0 # set door to its initial value
        time.sleep(1) # wait 1 second before the next action
    ## if the switch is closed and door does not equal 1
    if (io.input(door_pin)==False and door!=1):
        logger.log("Door","Close") # stream a message saying "Close"
        logger.flush() # send the message immediately

        door=1 # set door so that this loop won't act again until the
switch has been opened
```

Applications:

- To keep a check on garage doors of go-downs.
- Locker security.

Conclusion:

Using Initialstate, a workable prototype of an IoT project was created. It can be concluded that Raspberry pi, along with other sensors, can create many successful IoT projects.

References:

[1] http://blog.initialstate.com/pi-for-kids-door-sensor/
[2] https://www.initialstate.com/
[3] https://www.aliexpress.com/

Experiment 46: Upload data on Thingspeak via Raspberry Pi

Aim:

To get temperature and humidity information using DHT 11 and Raspberry Pi and upload it on a Thingspeak channel.

Objective:

To be able to create useful of Internet of things based projects.

Components Required:

1. Raspberry PI 2 & SD Card with Raspbian Operating System
2. USB Power Supply
3. USB Cable
4. Breadboard & Jumper Wires
5. 2 x DHT 11 sensors
6. 2 x 10K Resistors
7. 2 x Photocells
8. 2 x 1uF Capacitors

Procedure:

I. Prep the Raspberry Pi

If you have not done so, load Raspbian on your Raspberry PI. If you don't have a Raspberry PI, you can get one at Soldering Sunday that includes NOOBS pre-loaded on the MicroSD card or you can follow our guide to loading the Operating System for your Raspberry PI. Once your Raspberry Pi is up and running we need to setup Python to talk to the GPIO pins. The GPIO pins are our interface to the DHT11 Temp/Humidity sensor and the photocell. For a deeper look at the Raspberry Pi GPIO Pin's go to our GPIO Tutorial.

Configuring Python

Not all the libraries we need to make this project are pre-loaded on the Raspberry Pi. You will need the Adafruit GPIO Python library and the Adafruit DHT 11 library.

Use Adafruit's guide and library for setting up Python to communicate with the Raspberry Pi GPIO pins.

https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-gpio

You also need Adafruit's Python library for the DHT11 sensor, which you can find here:

https://learn.adafruit.com/dht-humidity-sensing-on-raspberry-pi-with-gdocs-logging/overview

II. Raspberry PI - Understanding GPIO Pins

Different versions of the Raspberry Pi have a different amount of GPIO pins. In early versions of the Raspberry Pi there were 26 pins and the more recent versions have 40 pins. Even though more pins were added, pins 1 through 26 are the same on all versions. When you look at a reference for Raspberry Pi GPIO pins you will will find several notations for each pin. Most commonly you will find one reference for the physical name of the pin (1 to 40) and the other for the GPIO Name (GPIO1, etc). The physical name is just that, the physical ordered number of the pin. With Python we will be using the GPIO reference for our pin identification. The GPIO Name is designated from the chip set and more commonly used in advanced projects. Referencing the wrong GPIO Pin number is very common and if you are not getting the results you expect when working with GPIO, double check the pin you are connected to and the pin you are referencing in your code.

III. Build the Circuit

Before connecting anything to your Raspberry PI, disconnect the power. Warning - you can destroy your Raspberry Pi with a short circuit from a wrong connection. Just be careful and double check everything before powering back on.
For this circuit we need to use the 3.3v out from the Raspberry Pi Pin 1 (do not use the 5v on Pin 2) and we need Ground (GND) of course. Connect these from the Pi to the Breadboard.
The DHT 11 has 4 Pins. Pin 1 is VCC, Pins 2 is Data, Pin 3 is NOT USED, Pin 4 is Ground.

- Connect DHT 11 Pin 1 to 3.3v
- Connect DHT 11 Pin 2 to Raspberry PI Pin 16/GPIO 23 and connect a 4.7 or 10k resistor from DHT 11 Pin 2 to DHT Pin 1
- Connect DHT 11 Pin 4 to Ground

The photo resistor has 2 pins

- Connect one pin to 3.3.v
- Connect the Other Pin to Raspberry Pi Pin 18/GPIO 24
- Connect a 1uF Capacitor to the same pin that the photo resistor is connected to on GPIO24. The Ground (White Stripe) side of the capacitor should go to Ground.
- Check your work against the attached Fritzing Diagram and the photos.

IV. Step 5: Setup Thingspeak for our IOT data

Our Python script is going to read data from the DHT11 sensor and the photoresistor and then publish the values of that data to our channel on Thingspeak. First we need to set it up.
Go to Thingspeak.com and create a free account or login to you existing account. Click on "My Channels" and then click on New Channel. Name your new channel and name the fields. The order of the fields is important later when we post data.
They can be in any order, but when you post the data the data you need to remember position.
You can decide if you want the channel to be public or not as well as publish information about it's location. This is all up to you and will not affect our code. You will also need the Write API key for the channel as it will be required to post data to the channel.

V. Create the Python Script

Upload the Python Script. You can copy paste to your IDLE session or download it to your Raspberry PI and save it in the /pi folder.
Before running it be sure to edit your API key from your ThingSpeak channel - edit this line:

```
#Setup our API and delay
myAPI = "***Insert Your API CODE HERE***"
```

VI. Start Your IOT Engine!

type LS to see the file listings and the myfirstIOT.py should be there. We need SUDO (administrative) rights for script to access the GPIO Pins, so run the script as follows:
sudo python myfirstIOT.py
You should see the base url printed with your API key and the next line will be data from the DHT 11.
Each line printed locally will display:
Temp (c) - Temp (f) - Humidity - Light (0=off / 1=on)
Then go check you Channel on Thingspeak, you should have data.

Source Code:

```
import time # so we can use "sleep" to wait between actions

import RPi.GPIO as io # import the GPIO library we just installed but call
it "io"

from ISStreamer.Streamer import Streamer # import the IS Streamer we just
installed but call it "Streamer"



## name the bucket and individual access_key

## the bucket_key will send all of our messages to the same place

## the access_key tells Initial State to send the messages to you

logger=Streamer(bucket_name="Locker
Protector",bucket_key="locker_protector",access_key="Your_Access_Key_Here")



## set GPIO mode to BCM

## this takes GPIO number instead of pin number

io.setmode(io.BCM)



## enter the number of whatever GPIO pin you're using

door_pin = 23



## use the built-in pull-up resistor

io.setup(door_pin, io.IN, pull_up_down=io.PUD_UP)  # activate input with
PullUp



## initialize door
```

```
door=0


## this loop will execute the if statement that is true
while True:
    ## if the switch is open
    if io.input(door_pin):
        logger.log("Door","Open") # stream a message saying "Open"
        logger.flush() # send the message immediately
        door=0 # set door to its initial value
        time.sleep(1) # wait 1 second before the next action
    ## if the switch is closed and door does not equal 1
    if (io.input(door_pin)==False and door!=1):
        logger.log("Door","Close") # stream a message saying "Close"
        logger.flush() # send the message immediately
        door=1 # set door so that this loop won't act again until the
switch has been opened
```

Applications:

- Control an ambient critical system remotely
- Farming
- Home automation

Conclusion:

Using Thingspeak, a workable prototype of an IoT project was created. It can be concluded that Raspberry pi, along with other sensors, can create many successful IoT projects.

References:

[1] https://cdn-learn.adafruit.com/downloads/pdf/dht.pdf
[2] https://thingspeak.com/
[3] http://www.instructables.com/id/Build-Your-First-IOT-with-a-Raspberry-Pi-DHT11-sen/?ALLSTEPS

Experiment 47: Communication between Raspberry Pi and Arduino: Wired

Aim:

To communicate between Raspberry Pi and Arduino via wired connection.

Objective:

To learn how to send data from Raspberry Pi to Arduino and vice versa.

Components Required:

1. Raspberry Pi and power supply adapter.
2. Screen to view the RPi desktop:
   - HDMI to VGA cable and Computer Monitor or
   - Touchscreen Display for RPi or
   - View the RPi screen remotely using VNC viewer.
3. Arduino and its USB cable.

Connections:

1. Connect the power supply micro USB cable and the HDMI port of the HDMI to VGA cable to the RPi.
2. Connect the VGA port of the HDMI to VGA cable to any computers monitor's VGA port.
3. That's all, now switch on the power supply.
4. If you want to view the RPi screen remotely using VNC server, follow the steps given in Reference [2].
5. Plug the USB cable of the Arduino in the USB port of the RPi.

Note: Do not use any mobile charger adapter to power on the RPi. Use 5V-2A for RPi 2 and 5V-2.5A for RPi 3.

Procedure:

8. Once your RPi is up and ready, you will need to install arduino IDE on the RPi. To do so, go to the RPi terminal and type the following:

   Command: sudo apt-get install arduino

9. You will now find Arduino Ide under programming section of the Rpi.

10. We need to install a package which allows python to communicate serially to Arduino via USB.

    Command : sudo apt-get install python-serial

Part A : Arduino Transmits, RPi receives.

1. Create a new python script and upload the source code given in Part A.
2. Go to the Arduino Ide and copy paste the following code given in Part A.
3. Upload the program on the arduino and run the python script.

Part B : RPI transmits, Arduino receives

1. Create a new python script and upload the source code given in Part B.
2. Go to the Arduino Ide and copy paste the following code given in Part B.
3. Upload the program on the arduino and run the python script.

Source Code:

Part A : Arduino Transmits, RPi receives.

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  //keep printing 0 to 11 on the Pi
  for (int i = 0 ; i < 11 ; i++) {
    Serial.println(i);
  };
}
```

```
import serial
ser=serial.Serial('/dev/ttyACM0',9600)

while 1:
   ser.readline()
```

Part B : RPI transmits, Arduino receives

```
import serial
ser=serial.Serial('/dev/ttyACM0',9600)
#keep sending 3 to arduino
ser.write('3')
```

```
void setup() {
  Serial.begin(9600);
}

void loop() {

  //Capture and print if Rpi sends data.
  if (Serial.available() > 0) {
    char incomming = Serial.read();
    Serial.println(incomming);
  };
}
```

Applications:

- Easy wired communication between Arduino and Pi which can be used t transfer sensor data and thereby, control any appliance of any organization or industry .

Conclusion:

Communication between Arduino and RPi is achieved using wires.

References:

[1] https://oscarliang.com/connect-raspberry-pi-and-arduino-usb-cable/
[2] https://www.raspberrypi.org/documentation/remote-access/vnc/README.md

Experiment 48: Communication between Raspberry Pi and Arduino: Wireless(Bluetooth)

Aim:

To communicate between Raspberry Pi and Arduino via Bluetooth wireless connection.

Objective:

To learn how to use Bluetooth to send data from Raspberry Pi to Arduino and vice versa .

Components Required:

1. Raspberry Pi and power supply adapter.
2. Screen to view the RPi desktop:
   - HDMI to VGA cable and Computer Monitor or
   - Touchscreen Display for RPi or
   - View the RPi screen remotely using VNC viewer .
3.Arduino and its USB cable.
4.HC-05 module.

Connections:

* HC-05 | Arduino
* TX | 5
* RX | 6

Please use Rpi 3 for this particular experiment!

1.Connect the power supply micro USB cable and the HDMI port of the HDMI to VGA cable to the RPi.
2.Connect the VGA port of the HDMI to VGA cable to any computers monitor's VGA port.
3.Thats all, now switch on the power supply.
4.If you want to view the RPi screen remotely using VNC server, follow the steps given in Reference [2].
5. Connect the LED negative to GND of Arduino and positive to pin 9 with a resistance valued between 220Ω – 1KΩ.

Note: Do not use any mobile charger adapter to power on the RPi. Use 5V-2A for RPi 2 and 5V-2.5A for RPi 3.

Procedure:

1. Once your Raspberry Pi is up and ready to use, You will need to download a package.

Go to RPi's terminal and type in the following command. You will require to connect your Pi to the internet for the following step.

sudo apt-get install python-serial

Once installed, you're ready to proceed.

Part A: Raspberry Pi transmits and Arduino receives.

1. Create a python file on the Pi and copy paste the source code of Part A given below.

2. Connect your Arduino Board and copy paste the source code of Part A given below.

3. Run both the programs.

4. Type 1 on the Rpi and see the led on pin 13 glow. Type 0 to see the Led on Pin 13 turn off.

Type q to quit the program.

Part B: Arduino transmits and Raspberry Pi receives.

1. Create a python file on the Pi and copy paste the source code of Part B given below.

2. Connect your Arduino Board and copy paste the source code of Part B given below.

3. Run both the programs.

4. The Python shell will keep on displaying 1 at an interval of 2 seconds!

Now, once you have accomplished the basics, you can send sensor data instead of sending random data and even control all the arduinos pins from the raspberry pi wirelessly.

Source Code:

Part A:

```
#Python code for Raspberry Pi Tx
import bluetooth

bd_addr = "98:D3:33:80:68:08" #the address of the HC-05 bluetooth sensor.
port = 1
sock = bluetooth.BluetoothSocket (bluetooth.RFCOMM)
sock.connect((bd_addr,port))

#Typing q will end the program.
#Typing 1 will glow the led on pin 13 on Arduino Board.
while 1:
      tosend = raw_input()
      if tosend != 'q':
            sock.send(tosend)
      else:
            break

sock.close()
```

```
/**
 * Arduino code to receive data from Rpi via Bluetooth
 * Arduino connection HC-05 connection:
 * HC-05  | Arduino
 * TX     | 5
 * RX     | 6
*/
#include <SoftwareSerial.h> //Serial library

SoftwareSerial bt(5,6);  //Bluetooth Tx to 5.
int LEDPin = 13; //LED PIN on Arduino
int btdata; // the data given from the RPi

void setup() {
  bt.begin(9600);
  bt.println ("Ready to  receive");
  pinMode (LEDPin, OUTPUT);
}

void loop() {
  if (bt.available()) {
    btdata = bt.read();
    if (btdata == '1') {
      digitalWrite (LEDPin, HIGH);
      bt.println ("LED ON!");
    }
    if (btdata == '0') {
      digitalWrite (LEDPin, LOW);
      bt.println ("LED OFF!");
    }
  }
  delay (100);
}
```

Part B:

```
import bluetooth
#Even though Rpi receives data,it has to initiate connection first.

target_name = "HC-05"
target_address = "98:D3:33:80:68:08" #the address of the HC-05 bluetooth sensor.
nearby_devices = bluetooth.discover_devices()

for bdaddr in nearby_devices:
    if target_name == bluetooth.lookup_name( bdaddr ):
        target_address = bdaddr
        break

if target_address is not None:
    print "found target bluetooth device with address ", target_address
else:
    print "could not find target bluetooth device nearby"

#bd_addr = "98:D3:33:80:68:08" #The address from the HC-05 sensor
port = 1
sock = bluetooth.BluetoothSocket (bluetooth.RFCOMM)
sock.connect((target_address,port))
print "connected to HC-05"

data = ""
while 1:
    try:
        data = sock.recv(1024)
        print data
        #data += sock.recv(1024)
        #data_end = data.find('\n')
        #if data_end != -1:
            #rec = data[:data_end]
            #print data
            #data = data[data_end+1:]
    except KeyboardInterrupt:
        break
sock.close()
```

```
/**Arduino code to transmit data to Rpi via Bluetooth
 * Arduino connection HC-05 connection:
 * HC-05  | Arduino
 * TX     | 5
 * RX     | 6
*/
#include <SoftwareSerial.h> //Serial library

SoftwareSerial bt(5, 6); //Bluetooth Tx to 5.
int btdata = 1; // the data given from arduino.Change accordingly.

void setup() {
  bt.begin(9600);
}

void loop() {
  //Keep on sending 1 with a delay of 2 seconds
```

```
  bt.print(btdata);
  delay (2000);
}
```

Applications:

- Send sensor data to the arduino from Raspberry Pi and vice versa.

Conclusion:

Information between Arduino and Raspberry Pi can be easily and wirelessly transmitted.

References:

[1] https://people.csail.mit.edu/albert/bluez-intro/x232.html

[2] https://www.raspberrypi.org/documentation/remote-access/vnc/README.md

[3] http://blog.whatgeek.com.pt/2015/09/bluetooth-communication-between-raspberry-pi-and-arduino/

Experiment 49: Matlab : Read data from external site, process it and tweet alerts

Aim:

To read data from external site, process it and tweet alert using Matlab via Thingspeak.

Objective:

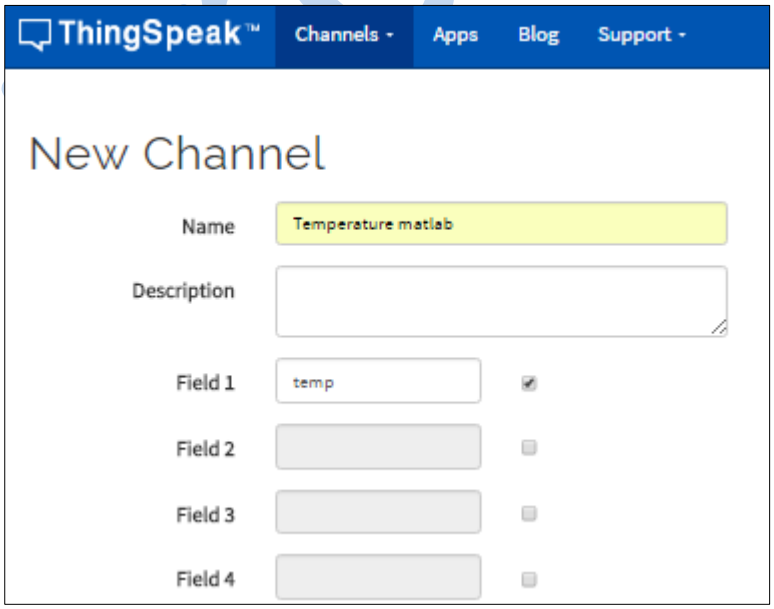
To be able to create a unique Internet of Things project using Thingspeak and understand and apply Matlab codes to achieve result.

Components Required:

1. Only Computer with Internet access.

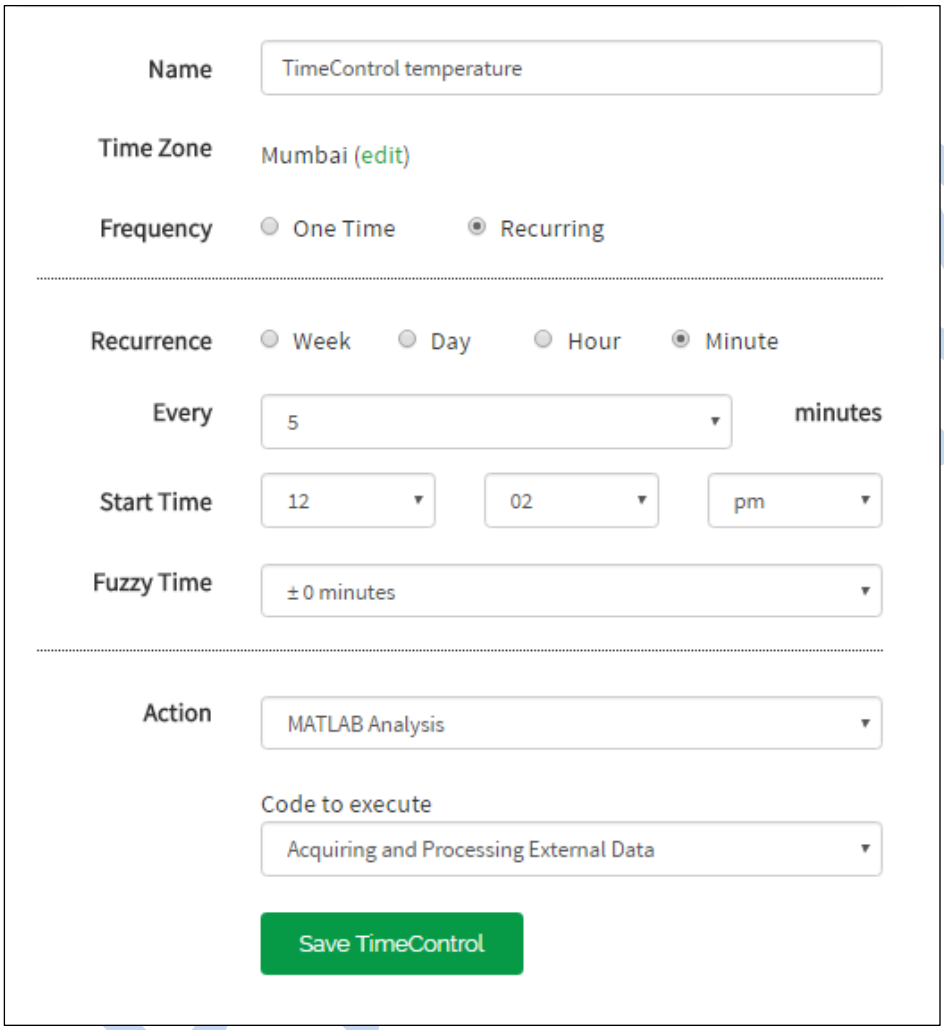
Procedure:

1. Go to Thingspeak and create an account. Link provided in Reference [1].
2. Create a new channel and fill in the details shown in Figure below and save it.



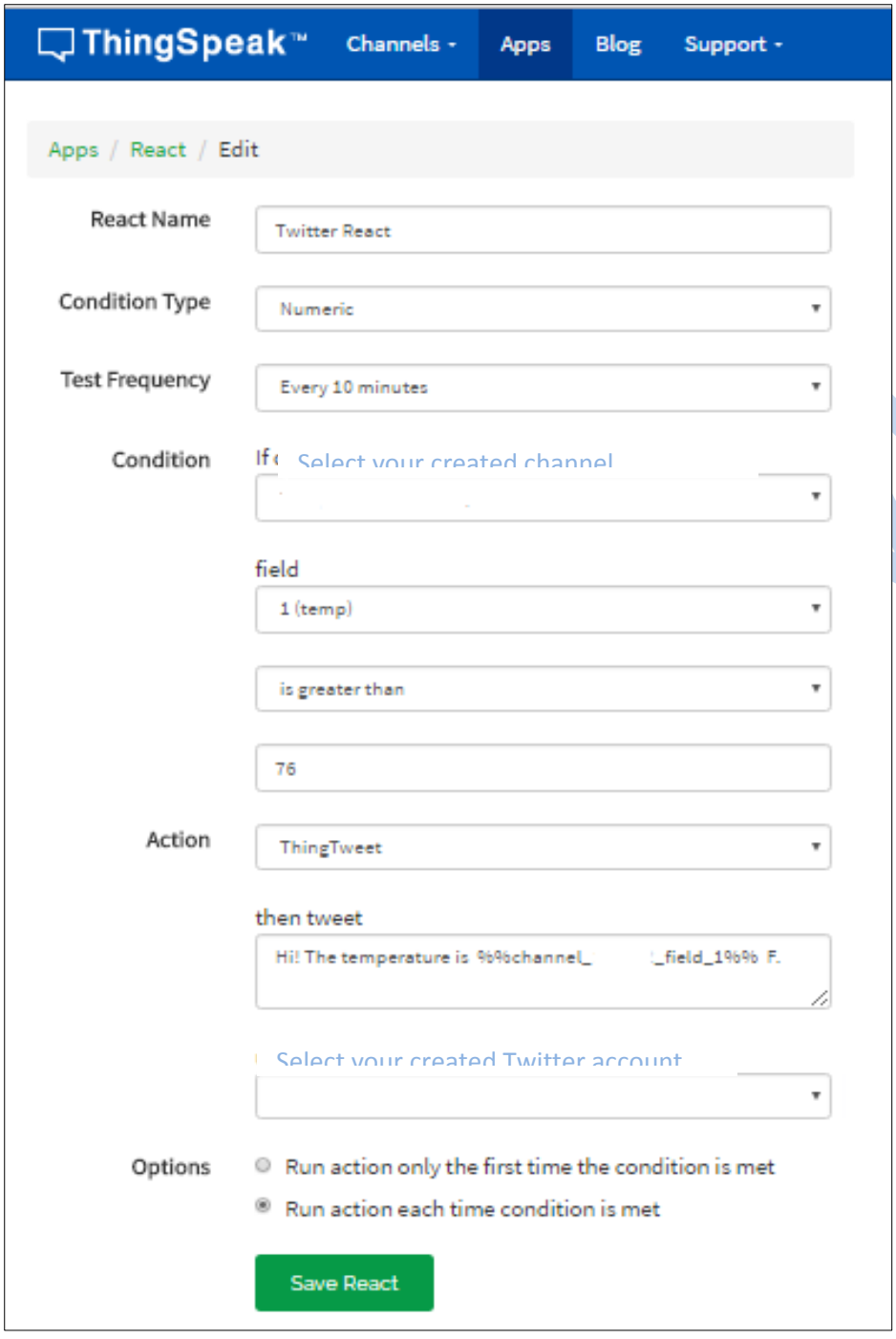
3. Next go to APPS, and click on *MATLAB analysis >> New >> Create.*

4. Name it as 'Acquiring and Processing External Data'. Enter the Matlab code given in the Source code section below.
5. In the code given, we first acquire data from an external site, in this case external channel whose is Link provided in reference [2].
6. We process it by taking the median of readings in the interval of every 5 minutes and then write it to our newly created channel.
7. Click on Time Control after saving the analysis, and fill in the details as shown below.



8. This will keep acquiring and writing data to your channel after processing it.
9. Next step is to tweet alert if the temperature has crossed a threshold value.
10. Create a new Twitter account for yourself. Next in Thingspeak click on *APPS >> ThingTweet >> Link Twitter Account.*

11. Once you've linked your Twitter account, Click on *APPS >> React >> New React*



12. In the space then tweet, enter the following by replacing xxxxx with your id:
Hi! The temperature is %%channel_xxxxx_field_1%% F.

13. When the value is greater than 76 F, you can view your tweet alert!

Source Code:

```
% Enter your MATLAB code below

range = median(thingSpeakRead(87179,'Fields',[1],'NumMinutes',5,...

  'URL','http://thingspeak.com/'));


% Write the depth to a new ThingSpeak channel


%Replace xxxxx with your channel ID.


thingSpeakWrite(xxxxx, range, 'WriteKey', 'BIVR5LDZYUIZEA61',...

 'URL', 'https://api.thingspeak.com');
```

Applications:

- Different parameters like wind velocity, temperature, dew point can be extracted from external site and alerts can be tweeted if threshold value is crossed.

Conclusion:

Various MATLAB apps like Analysis can be used to illustrate data and get comprehensive analysis and tweet alerts

References:

[6] https://thingspeak.com/
[7] http://makerzone.mathworks.com/producthardware-detail/real-timetide-gauge-to-tweet-tidal-alerts/
[3] https://www.hackster.io/hliang/thingspeak-weather-station-data-analysis-2877b0

Thank You !

Use this QR code to download IoT Course with Codes