Department of Computer Science and Engineering (IoT)

Course File

II B. Tech II Semester
Subject: Applications of IoT Lab

Code: A6907

Academic Year 2022-23
Regulations: MR 20

Malla Reddy Engineering College
(Autonomous)
Department of Computer Science and Engineering-IoT
(Approved by AICTE & Affiliated to JNTUH, Hyderabad)
Maisammaguda, Dhulapally(Post via Kompally) Secunderabad - 500100

Contents

INDEX

9.	Select at least 1 input sensor (not PIR) and 1 output device and make the RPi control the chosen output device in response to activity by the input device (e.g., a temperature sensor as input and two or more LEDs indicating the current temperature in binarycode).			
10.	Write a python program for client-server-based intruder detection system using mqtt application layer protocol.			
11.	Write an Arduino sketch to blink an LED Light for a particular interval of time using wireless communication protocol(LoRa).			
12.	**Case study:** • Assume that you are in a college, design and implement a IoT prototype to measure the amount of usage of water at a given location (take the location from user) on a day-to-day basis and send the information to Cloud. • Receive the above information from the sensors/ cloud and apply necessary algorithms to predict the amount of water being wasted at a particular location and also send a notification to the user.			

Lab In charge **HOD**

Malla Reddy Engineering College
(Autonomous)
Department of Computer Science and Engineering-IoT

Institute Vision

To be a premier centre of professional education and research, offering quality programs in a socio-economic and ethical ambience.

Institute Mission

1. **To impart knowledge of advanced technologies using state-of-the-art infrastructural facilities.**
2. **To inculcate innovation and best practices in education, training and research.**
3. **To meet changing socio-economic needs in an ethical ambience.**

Vision

To attain global standards in Computer Science and Engineering education, training, and research to meet the growing needs of the industry with socio-economic and ethical considerations.

Mission

To impart quality education and research to undergraduate and postgraduate students in Computer Science and Engineering.

To encourage innovation and best practices in Computer Science and Engineering utilizing state-of-the-art facilities.

To develop entrepreneurial spirit and knowledge of emerging technologies based on ethical values and social relevance.

Programme Educational Objectives (PEOs)

PEO I:

Graduates to promote collaboration with the reputed industries to have the best careers.

PEO II:

Graduates will demonstrate technical skills, competency in IOT and exhibit team management capability with proper communication in a job environment.

PEO III:

To nurture the sense of creativity and innovation to adopt the socio-economic related activities Graduates will interact with their peers in other disciplines in industry and society and contribute to the economic growth of the country.

PEO IV:

To enable graduates to emerge as independent entrepreneurs and future leaders.

PEO V:

To provide the sate- of- the art facilities to forge the students in industry-ready IoT system development.

Programme Outcomes

PO	Programme Outcomes
PO 1	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO 2	Problem analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO 3	Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO 4	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO 5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO 6	The engineer and society: Apply reasoning informed by the contextual knowledge to societal, health, safety, legal and cultural issues and the consequent responsibilities relevant professional engineering practice.
PO 7	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO 8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO 9	Individual and team work: Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.
PO 10	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as,being able to comprehend and write effective reports and design
PO 11	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO 12	Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Bloom's Taxonomy Triangle

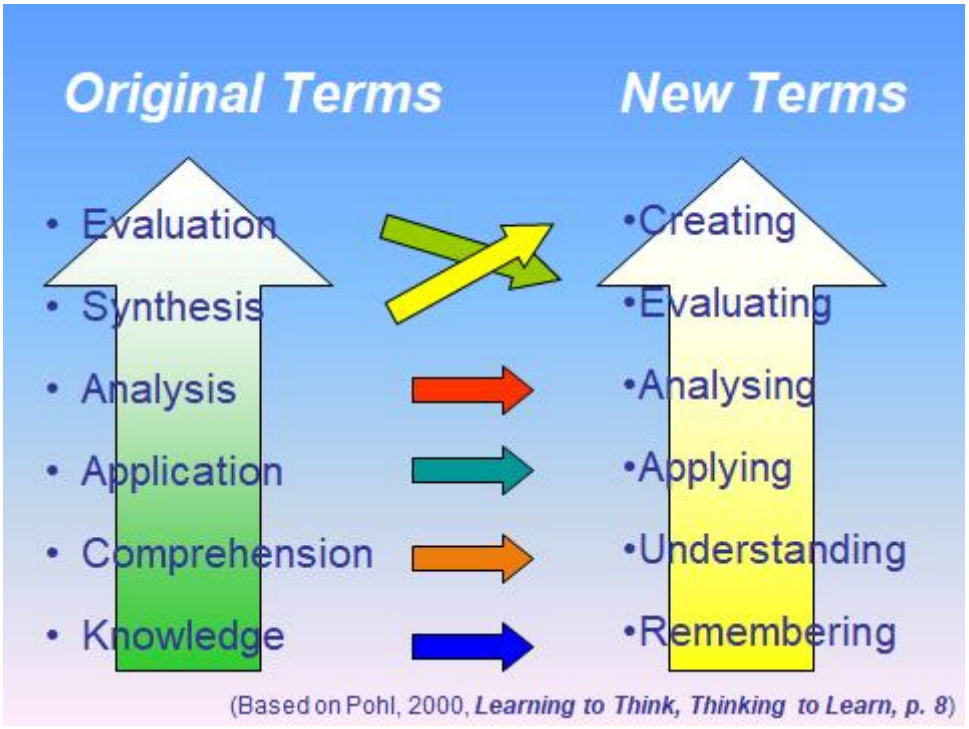
Overview

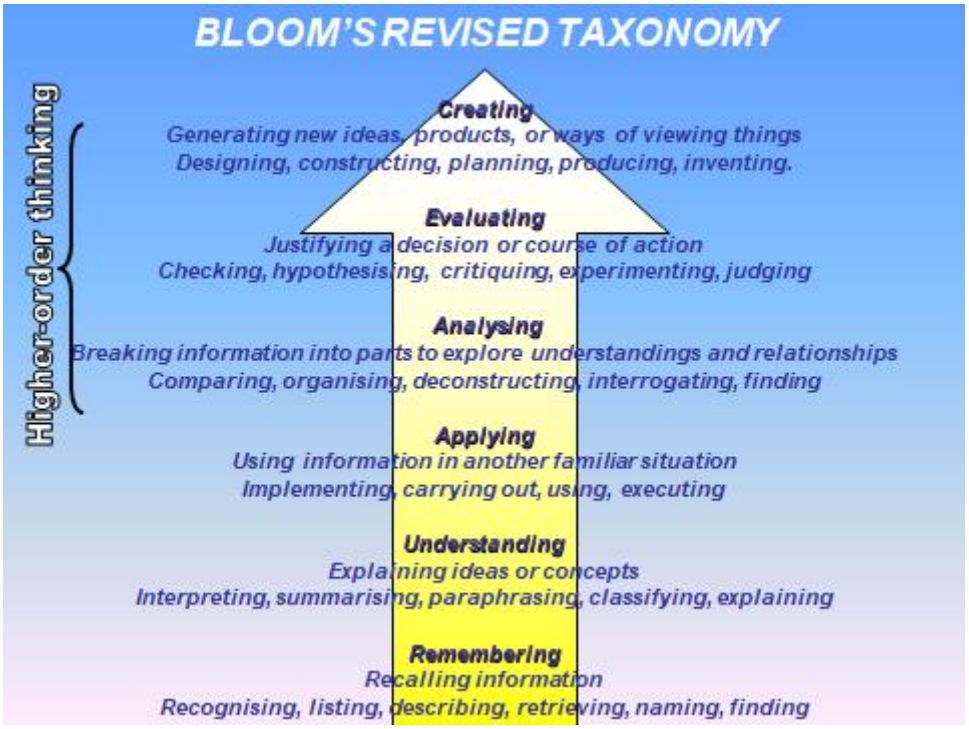
- Bloom's Taxonomy and higher-order thinking
- Take a walk down memory lane
- Investigate the Revised Taxonomy
  - New terms
  - New emphasis
- Explore each of the six levels
- See how questioning plays an important role within the framework (oral language)
- Use the taxonomy to plan a unit
- Look at an integrated approach
- Begin planning a unit with a *SMART Blooms* Planning Matrix

Bloom's Revised Taxonomy

- Taxonomy of Cognitive Objectives
- 1950s- developed by Benjamin Bloom
- Means of expressing qualitatively different kinds of thinking
- Adapted for classroom use as a planning tool
- Continues to be one of the most universally applied models
- Provides a way to organise thinking skills into six levels, from the most basic to the higher order levels of thinking
- 1990s- Lorin Anderson (former student of Bloom) revisited the taxonomy
- As a result, a number of changes were made

(Pohl, 2000, *Learning to Think, Thinking to Learn, pp. 7-8)*


Original Terms
New Terms
• Evaluation
• Synthesis
• Analysis
• Application
• Comprehension
• Knowledge
•Creating
•Evaluating
•Analysing
•Applying
•Understanding
•Remembering
(Based on Pohl, 2000, Learning to Think, Thinking to Learn, p. 8)


BLOOM'S REVISED TAXONOMY
Higher-order thinking
Creating
Generating new ideas, products, or ways of viewing things
Designing, constructing, planning, producing, inventing.
Evaluating
Justifying a decision or course of action
Checking, hypothesising, critiquing, experimenting, judging
Analysing
Breaking information into parts to explore understandings and relationships
Comparing, organising, deconstructing, interrogating, finding
Applying
Using information in another familiar situation
Implementing, carrying out, using, executing
Understanding
Explaining ideas or concepts
Interpreting, summarising, paraphrasing, classifying, explaining
Remembering
Recalling information
Recognising, listing, describing, retrieving, naming, finding

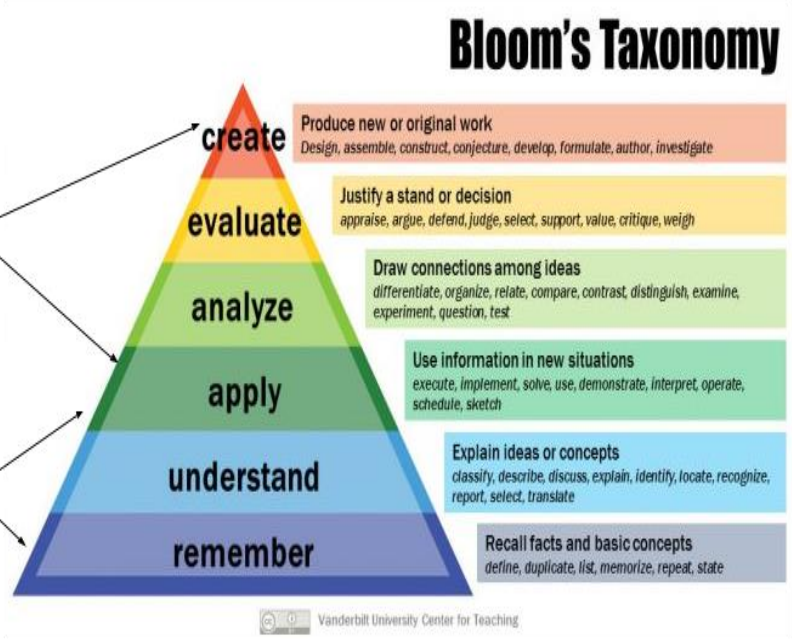
REVISED Bloom's Taxonomy Action Verbs

Definitions	I. Remembering	II. Understanding	III. Applying	IV. Analyzing	V. Evaluating	VI. Creating
Bloom's Definition	Exhibit memory of previously learned material by recalling facts, terms, basic concepts, and answers.	Demonstrate understanding of facts and ideas by organizing, comparing, translating, interpreting, giving descriptions, and stating main ideas.	Solve problems to new situations by applying acquired knowledge, facts, techniques and rules in a different way.	Examine and break information into parts by identifying motives or causes. Make inferences and find evidence to support generalizations.	Present and defend opinions by making judgments about information, validity of ideas, or quality of work based on a set of criteria.	Compile information together in a different way by combining elements in a new pattern or proposing alternative solutions.
Verbs	• Choose • Define • Find • How • Label • List • Match • Name • Omit • Recall • Relate • Select • Show • Spell • Tell • What • When • Where • Which • Who • Why	• Classify • Compare • Contrast • Demonstrate • Explain • Extend • Illustrate • Infer • Interpret • Outline • Relate • Rephrase • Show • Summarize • Translate	• Apply • Build • Choose • Construct • Develop • Experiment with • Identify • Interview • Make use of • Model • Organize • Plan • Select • Solve • Utilize	• Analyze • Assume • Categorize • Classify • Compare • Conclusion • Contrast • Discover • Dissect • Distinguish • Divide • Examine • Function • Inference • Inspect • List • Motive • Relationships • Simplify • Survey • Take part in • Test for • Theme	• Agree • Appraise • Assess • Award • Choose • Compare • Conclude • Criteria • Criticize • Decide • Deduct • Defend • Determine • Disprove • Estimate • Evaluate • Explain • Importance • Influence • Interpret • Judge • Justify • Mark • Measure • Opinion • Perceive • Prioritize • Prove • Rate • Recommend • Rule on • Select • Support • Value	• Adapt • Build • Change • Choose • Combine • Compile • Compose • Construct • Create • Delete • Design • Develop • Discuss • Elaborate • Estimate • Formulate • Happen • Imagine • Improve • Invent • Make up • Maximize • Minimize • Modify • Original • Originate • Plan • Predict • Propose • Solution • Solve • Suppose • Test • Theory

Anderson, L. W., & Krathwohl, D. R. (2001). A taxonomy for learning, teaching, and assessing, Abridged Edition. Boston, MA: Allyn and Bacon.

Focus of individual time in traditional model

Focus of class time in traditional model



The Graduate Attributes for UG Engineering Student

1. **Engineering Knowledge:** apply Knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

2. **Problem Analysis:** Identify, formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

3. **Design / Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.

4. **Conduct investigations of complex problems** using research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.

5. **Modern Tool Usage:** Select and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modeling, to broadly-defined engineering activities, with an understanding of the limitations.

6. **The Engineer and society:** Demonstrate understanding of the societal, health, safety, legal and cultural issues, and the consequent responsibilities relevant to engineering technology practice.

7. **Environment and sustainability:** Understand the impact of engineering/technology solutions in societal and environmental context, and demonstrate knowledge of, and need for sustainable development.

8. **Ethics:** Understand and commit to professional ethics and responsibilities and norms of engineering technology practice.

9. **Individual and Team work:** Function effectively as an individual, and as a member or leader in diverse technical teams.

10. **Communication:** Communicate effectively on Broadly-defined engineering activities with the engineering community and with society at large, by being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project Management and Finance:** Demonstrate knowledge and understanding of engineering management principles and apply the same to one's own work, as a member and leader in a team and to manage projects in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the ability to engage in independent and life-long learning in specialized technologies.

Course Objectives:

Develop ability to

1. Assess the vision and introduction of IoT and understanding how M2M is connected to internet of things
2. Identify the appropriate Hardware and software components of IoT for communication
3. Gain knowledge on Cloud Storage models, web servers and how to integrate device, data and cloud management framework for IoT.
4. Learn the concepts of various data analytics and operational technology security with IoT.
5. Understand advanced and emerging concepts fog computing and Edge computing-IoT

Course Outcomes (COs):

After completion of the course, student would be able to

CO1: Interpret the vision of IoT from a global context, compare and contrast M2M and IoT Technology

CO2: Relate the appropriate Hardware and software components of IoT for providing the communication among the devices

CO3: Implement device, data and cloud management services for IoT applications.

CO4: Explore various data analytical techniques and operational security for IoT applications.

CO5: Comprehend the need of Fog Computing and Edge Computing-IoT

CO-PO,PSOMapping

(3/2/1indicates strengthofcorrelation)3-Strong,2-Medium,1-Weak

COs	ProgrammeOutcomes(POs)												PSOs		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO 8	PO9	PO10	PO1 1	PO1 2	PSO 1	PSO 2	PSO 3
CO1	3	2	1								1	1	1		
CO2	3	2										1	2		
CO3	2	1										1	1		

2020-21 Onwards (MR-20)	MALLA REDDY ENGINEERING COLLEGE (Autonomous)	B.Tech. IV Semester		
Code: A6903	Internet of Things Fundamentals Lab	L	T	P
Credits: 1.5		-	-	3

List of Programs:

1. Getting Started with IoT (Arduino).
2. Write an Arduino sketch to blink an LED Light for a particular interval oftime.
3. Write an Arduino sketch to measure the distance (in cms) of a certainobject.
4. Write an Arduino sketch to
   i. Blink an LED and a buzzer if the distance measured is less than a threshold value
   ii. Illustrate the working of PIR Sensor with anexample.
   iii. Illustrate the IR and DHT Sensor.
5. Write an Program to send the humidity and temperature data to Cloud(ThingSpeak)
6. Write a program to alert the user through SMS and Email notification if humidity is greater than a threshold value using IFTTT and Thingspeak cloud.
7. Write a Python program that blinks an LED at a rate of 3 second ON, 1 secondOFF
8. Connect a PIR sensor to the GPIO pins of the Raspberry Pi. Perform measurements to determine the range of the sensor, i.e., start with a small distance (e.g., a few inches) and see if the motion sensor responds. Repeat these for increasing distances until the sensor stops responding. Report the measured distance.
9. Select at least 1 input sensor (not PIR) and 1 output device and make the RPi control the chosen output device in response to activity by the input device (e.g., a temperature sensor as input and two or more LEDs indicating the current temperature in binarycode).
10. Write a python program for client-server-based intruder detection system using mqtt application layerprotocol
11. Write an Arduino sketch to blink an LED Light for a particular interval of time using wireless communication protocol(LoRa)

Case study:

1. Assume that you are in a college, design and implement a IoT prototype to measure the amount of usage of water at a given location (take the location from user) on a day-to-day basis and send the information toCloud.
2. Receive the above information from the sensors/ cloud and apply necessary algorithms to predict the amount of water being wasted at a particular location and also send a notification to theuser.

1. Introduction Getting Started with IoT (Arduino).

Arduino is a platform that makes it easy for you to build projects using electronics. Arduino can also help you easily build IoT projects in two ways: Using traditional Arduino boards and attaching communication breakout modules (like Bluetooth, WiFi, LoRA, GSM, etc) to them. Arduino is a micro controller that can be connected to one or more sensors and help you capture the data or information and then pass it on to processor.

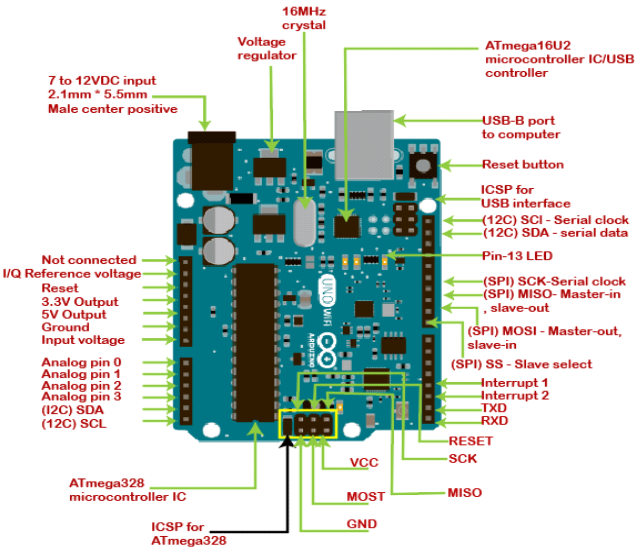
Features of Aurdino:

- Open source based electronic programmable board(micro controller) and software(IDE).
- Accepts analog and digital signals as input and gives desired output.
- No extra hardware required to load a program into the controller board

Arduino UNO:

Feature	Value
Clock speed	16MHz
Operating Voltage	5V
Digital I/O	14
Analog Input	6
PWM	6
UART	1

- interface USB via ATMega16U2



Arduino IDE:

Arduino IDE is an open source that is used to program the arduino controller board. It is based on variations of the C and C++ programming language. It can be downloaded from arduino's official website and installed into PC.

Download and install the Arduino software (Arduino IDE 1.8.5).

Go to the Arduino website and click the download link to go to the download page. After downloading, locate the downloaded file on the computer and extract the folder from the downloaded file. Copy the folder to a suitable place such as your desktop.

Set Up:

- Power the board by connecting it to a PC via USB cable
- Launch the Arduino IDE
- Set the board type and the port for the board
- Tools-->Board-->select your board
- Tools-->Port-->select your port

Arduino IDE Overview:

- Program coded in Arduino IDE is called a SKETCH
- To create a new sketch
- File-->New
- To open an existing sketch
- File-->open
- There are some basic ready-to-use sketches available in the EXAMPLES section
  File-->Examples-->select any program
- Verify: Checks the code for compilation errors
- Upload: Uploads the final code to the controller board
- New: Creates a new blank sketch with basic structure
- Open: Opens an existing sketch
- Save: Saves the current sketch
- Serial Monitor: Opens the serial console
  All the data printed to the console are displayed here

Sketch Structure:

A sketch can be divided into two parts:

Setup()

Loop()

The function setup() is the point where the code starts, just like the main() function in c and c++ . I/O variables, pin modes are initialized in the Setup() function. Loop() function, as the name suggests, iterates the specified task in the program.

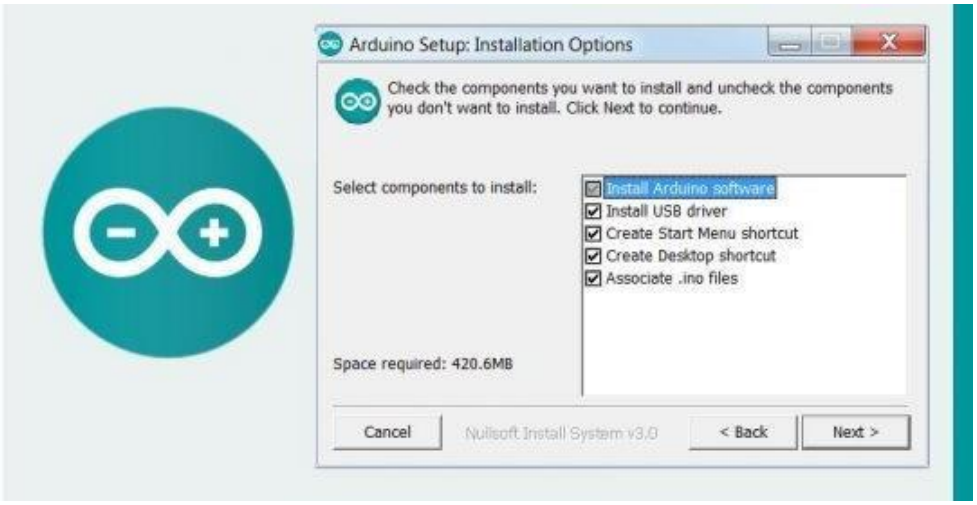
Aim: Familiarization with Arduino/Raspberry Pi and perform necessary software installation

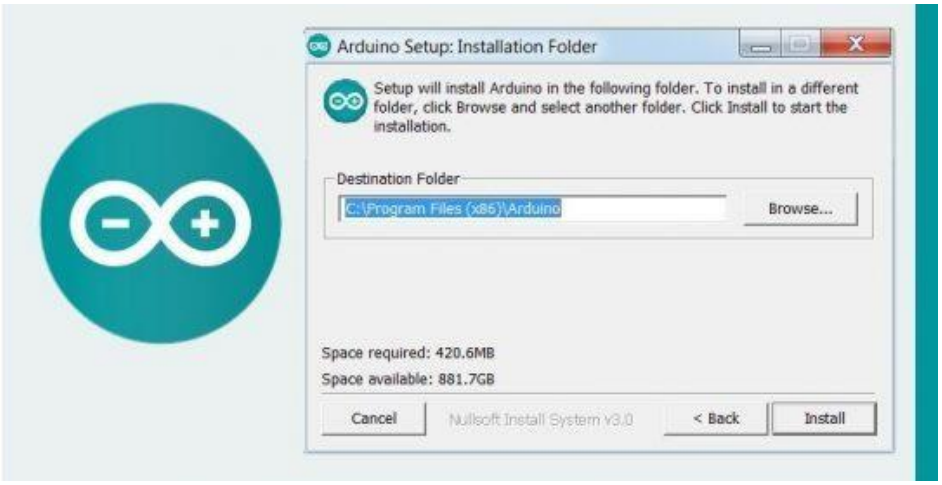
- Arduino is a platform that makes it easy for you to build projects using electronics.
- IoT is a way of using electronics - to make electronic modules talk to each other remotely and wirelessly (often using a Cloud) to solve problems.
- Now, Arduino can also help you easily build IoT projects in two ways: Using traditional Arduino boards and attaching communication breakout modules (like nRF Bluetooth, WiFi, LoRA, GSM, etc) to them.
- Arduino is a micro controller that can be connected to one or more sensors and help you capture the data or information and then pass it on to processor. If you know the full stack of IoT then you should also look at Raspberry.
- RaspPi is a microprocessor so the basic difference between Arduino and RasPi is that RaspPi is controller plus processor and Arduino is just a micro controller.
- They suit the need for different use cases. You can easily read online about this both.

Download and install the Arduino software (Arduino IDE 1.8.5)

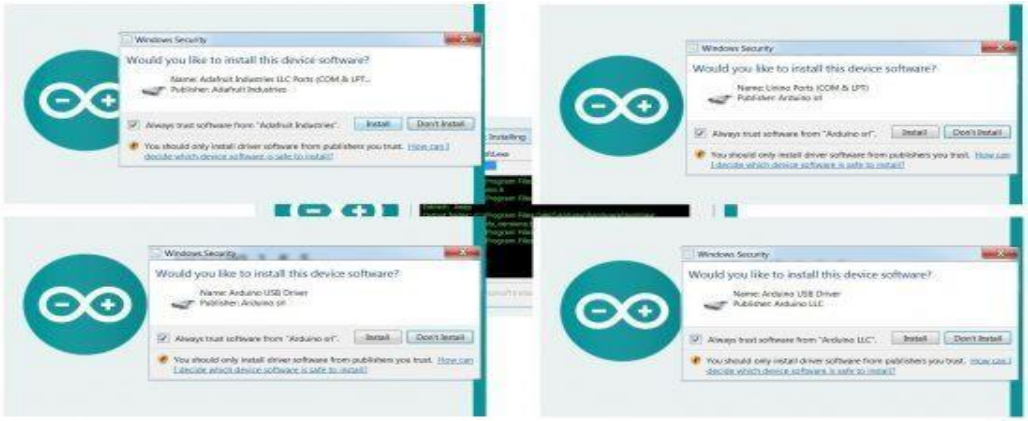
- Go to the Arduino website and click the download link to go to the download page.
- After downloading, locate the downloaded file on the computer and extract the folder from the downloaded file. Copy the folder to a suitable place such as your desktop.

Read the Arduino License agreement and click the “I Agree” button.





The Arduino software will start to install.



Running the Arduino IDE Software

2. Write an Arduino sketch to blink an LED Light for a particular interval of time

AIM: To interface LED/Buzzer with Arduino/Raspberry Pi and write a program to turn ON LED for 1 sec after every 2 seconds

Hardware Requirements:
1. 1x Breadboard
2. 1x Arduino Uno R3
3. 1x RGB LED
4. 1x 330Ω Resistor
5. 2x Jumper Wires

Procedure:
Connect the Arduino board to your computer using the USB cable.
Set the pin-mode as LED output
Set the pin-mode as Buzzer output.
Set the delay time for output
Set the digital pin-mode on.
Set the digital pin-mode off

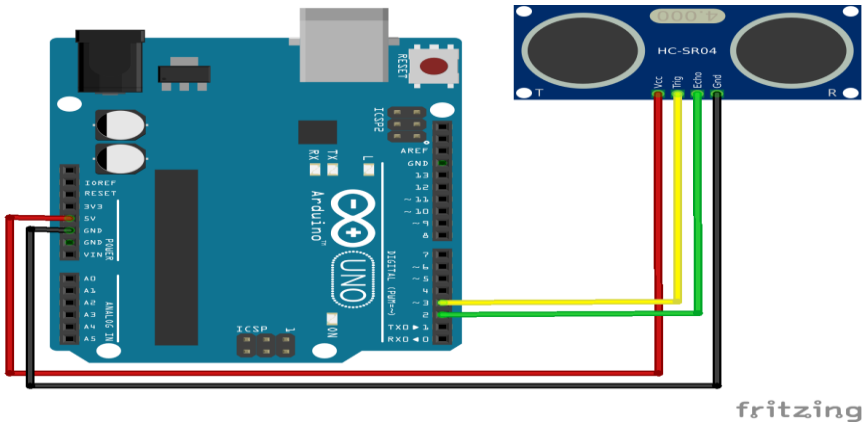
Source Code:

```
int led=13;
int buzzer=11;
void setup()
{
pinMode(13, OUTPUT);
```

```
    pinMode(11, OUTPUT);
    }
void loop()
{
digitalWrite(13, HIGH);
digitalWrite(11, HIGH);
delay(1000);
digitalWrite(13, LOW);
digitalWrite(11, LOW);
delay(2000);
}
```

3. **Write an Arduino sketch to measure the distance (in cms) of a certain object**

Ultrasonic Sensor HC-SR04 is a sensor that can measure distance.
It emits an ultrasound at 40 000 Hz (40kHz) which travels through the air and if there is an object or obstacle on its path It will bounce back to the module.
Considering the travel time and the speed of the sound you can calculate the distance.
The configuration pin of HC-SR04 is VCC (1), TRIG (2), ECHO (3), and GND (4).
The supply voltage of VCC is +5V and you can attach TRIG and ECHO pin to any Digital I/O in your Arduino Board.
In order to generate the ultrasound we need to set the Trigger Pin on a High State for 10 µs.
That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received in the Echo Pin.
The Echo Pin will output the time in microseconds the sound wave traveled.
The materials that we need to make this project:
1. Arduino UNO R3 CH340 (you can use any Arduino Boards)
2. Ultrasonic Sensor HC-SR04
3. Jumper Wires
4. Breadboard



Steps :

- First do the wiring as shown in the picture
- Open Arduino IDE Software and write down your code, or download the code below and open it
- Choose your own Arduino board (in this case Arduino Uno), by selecting Tools > Board > Arduino/Geniuno Uno
- Choose your COM Port (usually it appears only one existing port), Tools > Port > COM.. (If there are more than one ports, try it one by one)
- Upload your code by pressing Ctrl + U or Sketch > Upload
- To display the measurement data you can use Serial Monitor by pressing Ctrl + Shift + M (make sure that the baudrate speed is 9600)

SKETCH:

```
#define echoPin 2
#define trigPin 3
long duration;
int distance;
int distance;
void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
  Serial.println("Ultrasonic Sensor HC-SR04 Test");
  Serial.println("with Arduino UNO R3");
}
void loop() {
    digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
   duration = pulseIn(echoPin, HIGH);
   distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
}
```

4. Write an Arduino sketch to

1. **Blink an LED and a buzzer if the distance measured is less than a threshold value**
2. **Illustrate the working of PIR Sensor with an example.**
3. **Illustrate the IR and DHT Sensor.**

i.)

Code :

```
int sound = 150;
void setup() {
Serial.begin (9600);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
pinMode(led, OUTPUT);
pinMode(led2, OUTPUT);
pinMode(led3, OUTPUT);
pinMode(led4, OUTPUT);
pinMode(led5, OUTPUT);
pinMode(led6, OUTPUT);
pinMode(buzzer, OUTPUT);
}
void loop() {
long duration, distance;
 digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
```

```
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = (duration/2) / 29.1;
 if (distance <= 30) {
        digitalWrite(led, HIGH);
        sound = 250;
}
 else {
        digitalWrite(led,LOW);
 }
  if (distance < 25) {
        digitalWrite(led2, HIGH);
        sound = 260;
}
  else {
        digitalWrite(led2, LOW);
  }
  if (distance < 20) {
        digitalWrite(led3, HIGH);
        sound = 270;
}
  else {
        digitalWrite(led3, LOW);
  }
  if (distance < 15) {
        digitalWrite(led4, HIGH);
        sound = 280;
}
  else {
        digitalWrite(led4,LOW);
  }
 if (distance < 10) {
        digitalWrite(led5, HIGH);
        sound = 290;
}
  else {
        digitalWrite(led5,LOW);
  }
 if (distance < 5) {
        digitalWrite(led6, HIGH);
        sound = 300;
}
  else {
        digitalWrite(led6,LOW);
  }

  if (distance > 30 || distance <= 0){
        Serial.println("Out of range");
```

```
        noTone(buzzer);
  }
  else {
        Serial.print(distance);
        Serial.println(" cm");
        tone(buzzer, sound);
  }
  delay(500);
}
```

ii)

SKETCH:

```
int calibrationTime = 30;
long unsigned int lowIn;
long unsigned int pause = 5000;
boolean lockLow = true;
boolean takeLowTime;
int pirPin = 12;
int ledPin = 13;
void setup()
{
Serial.begin(9600);
pinMode(pirPin, INPUT);
pinMode(ledPin, OUTPUT);
digitalWrite(pirPin, LOW);
Serial.print("calibrating sensor ");
for(int i = 0; i < calibrationTime; i++)
{
Serial.print(".");
delay(1000);
 }
if(digitalRead(pirPin) == LOW)
{
Serial.println(" done");
 Serial.println("SENSOR ACTIVE");
delay(50);
 }
void loop()
{
if(digitalRead(pirPin) == HIGH)
{
digitalWrite(ledPin, HIGH);
if(lockLow)
{
lockLow = false;
 Serial.println("---");
Serial.print("motion detected at ");
Serial.print(millis()/1000);
```

```
Serial.println(" sec");
delay(50);
}
takeLowTime = true;
}
digitalWrite(ledPin, LOW);
if(takeLowTime){ lowIn = millis();
takeLowTime = false;
 }
if(!lockLow && millis() - lowIn > pause)
{
lockLow = true;
Serial.print("motion ended at ");
Serial.print((millis() - pause)/1000);
Serial.println(" sec"); delay(50);
}
}
}
```

iii)

SKETCH:

```
#include<DHT.h>;
        DHT dht(8, DHT22);
        float humidity;
        float temperature;
        void setup()
        {
        Serial.begin(9600);
        dht.begin();
        }
        void loop()
        {
        humidity=dht.readHumidity();
        temperature=dht.readTemperature();
        Serial.print("Humidity:");
        Serial.print(humidity);
        Serial.print("%, Temperature:");
        Serial.print(temperature);
        Serial.println("Celsius");
        delay(2000);
        }
```

5. Write an Program to send the humidity and temperature data to Cloud(ThingSpeak)

Code :

```
#include <DHT.h>
#include <ESP8266WiFi.h>
String apiKey = "APIKEY";
const char *ssid =  "Username";
const char *pass =  "Mrec@123";
```

```
const char* server = "api.thingspeak.com";
#define DHTPIN 0
DHT dht(DHTPIN, DHT11);
WiFiClient client;
void setup()
{
      Serial.begin(115200);
      delay(10);
      dht.begin();
      Serial.println("Connecting to ");
      Serial.println(ssid);
      WiFi.begin(ssid, pass);
     while (WiFi.status() != WL_CONNECTED)
    {      delay(500);
          Serial.print(".");
    }
     Serial.println("");
     Serial.println("WiFi connected");
}
void loop()
{
     float h = dht.readHumidity();
     float t = dht.readTemperature();
           if (isnan(h) || isnan(t))
              {
                 Serial.println("Failed to read from DHT sensor!");
                  return;
              }
          if (client.connect(server,80))   //   "184.106.153.149" or api.thingspeak.com
                {
                     String postStr = apiKey;
                      postStr +="&field1=";
                      postStr += String(t);
                      postStr +="&field2=";
                      postStr += String(h);
                      postStr += "\r\n\r\n";
                      client.print("POST /update HTTP/1.1\n");
                      client.print("Host: api.thingspeak.com\n");
                      client.print("Connection: close\n");
                      client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
                      client.print("Content-Type: application/x-www-form-urlencoded\n");
                      client.print("Content-Length: ");
                      client.print(postStr.length());
                      client.print("\n\n");
                      client.print(postStr);
                      Serial.print("Temperature: ");
                      Serial.print(t);
                      Serial.print(" degrees Celcius, Humidity: ");
                      Serial.print(h);
                      Serial.println("%. Send to Thingspeak.");
                 }
        client.stop();
         Serial.println("Waiting...");
  // thingspeak needs minimum 15 sec delay between updates
  delay(1000);
```

}

6. Write a program to alert the user through SMS and Email notification if humidity is greater than a threshold value using IFTTT and Thingspeak cloud.

Need to connect ThingSpeak and IFTTT no need to write extra code here

7. Write a Python program that blinks an LED at a rate of 3 second ON, 1 second OFF

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW)
while True:
   GPIO.output(8, GPIO.HIGH)
   sleep(1)
   GPIO.output(8, GPIO.LOW)
   sleep(1)
import RPi.GPIO as GPIO
from time import sleep
  GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW)
value to low (off)
while True:
 GPIO.output(8, GPIO.HIGH)
 sleep(1)
 GPIO.output(8, GPIO.LOW)
 sleep(1)
```

8. **Connect a PIR sensor to the GPIO pins of the Raspberry Pi. Perform measurements to determine the range of the sensor, i.e., start with a small distance (e.g., a few inches) and see if the motion sensor responds. Repeat these for increasing distances until the sensor stops responding. Report the measured distance.**

```
import RPi.GPIO as GPIO
import time

pir_sensor = 11
piezo = 7

GPIO.setmode(GPIO.BOARD)

GPIO.setup(piezo,GPIO.OUT)

GPIO.setup(pir_sensor, GPIO.IN)

current_state = 0
try:
   while True:
     time.sleep(0.1)
      current_state = GPIO.input(pir_sensor)
      if current_state == 1:
         print("GPIO pin %s is %s" % (pir_sensor, current_state))
         GPIO.output(piezo,True)
         time.sleep(1)
```

```
            GPIO.output(piezo,False)
            time.sleep(5)
except KeyboardInterrupt:
   pass
finally:
   GPIO.cleanup()
```

OR

Connect a PIR sensor to the GPIO pins of the Raspberry Pi. Perform measurements to determine

```
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11,GPIO.IN)
GPIO.setup(3,GPIO.OUT)
GPIO.output(3,GPIO.LOW)
while True:
if(GPIO.input(11)):
GPIO.output(3,GPIO.HIGH)
print("Intruder Detected!!")
 else:
GPIO.output(3,GPIO.LOW)
print("NO Intruder")
```

9. **Select at least 1 input sensor (not PIR) and 1 output device and make the RPi control the chosen output device in response to activity by the input device (e.g., a temperature sensor as input and two or more LEDs indicating the current temperature in binary code).**

```
import sys
import Adafruit_DHT
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8,GPIO.OUT)
GPIO.setup(10,GPIO.OUT)
GPIO.setup(12,GPIO.OUT)
GPIO.output(8,GPIO.LOW)
GPIO.output(10,GPIO.LOW)
GPIO.output(12,GPIO.LOW)
while True:
humidity, temperature = Adafruit_DHT.read_retry(11, 4)
#4 means pin number 7 in raspberry pi
print('Temp: {0:0.1f} C Humidity: {1:0.1f} %'.format(temperature, humidity))
temp_int = int(temperature)
if temp_int<20:
GPIO.output(8,GPIO.HIGH)
GPIO.output(10,GPIO.LOW)
GPIO.output(12,GPIO.LOW)

elif temp_int>20 and temp_int < 30:
GPIO.output(10,GPIO.HIGH)
GPIO.output(8,GPIO.LOW)
```

```
GPIO.output(12,GPIO.LOW)

elif temp_int >= 30:
GPIO.output(12,GPIO.HIGH)
GPIO.output(10,GPIO.LOW)
GPIO.output(8,GPIO.LOW)
```

10. Write a python program for client-server-based intruder detection system using mqtt application layer protocol.

Client.py

```
# Import socket module
import socket
import RPi.GPIO as GPIO
from time import sleep
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8,GPIO.OUT)
GPIO.output(8,GPIO.LOW)

# Create a socket object
s = socket.socket()
# Define the port on which you want to connect
port = 8787
# connect to the server on local computer
s.connect(('192.168.21.232', port))
# receive data from the server and decoding to get the string.
print (s.recv(1024).decode())
while True:
if s.recv(1024).decode()=='1':
print('intruder detected')
GPIO.output(8,GPIO.HIGH)
sleep(10)
else:
GPIO.output(8,GPIO.LOW)
print('intruder not detected')
# close the connection
s.close()
```

Server.py

```
import socket
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8,GPIO.IN)
# next create a socket object
s = socket.socket()
print ("Socket successfully created")
# reserve a port on your computer in our
# case it is 12345 but it can be anything
port = 8787
```

```
# Next bind to the port
# we have not typed any ip in the ip field
# instead we have inputted an empty
# this makes the server listen to
requests
# coming from other computers on the network
s.bind(('', port))
print ("socket binded to %s" %(port))
#
put the socket into listening mode
s.listen(5)
print ("socket is listening")
# a forever loop until we interrupt it or
# an error occurs
while True:
# Establish connection with client.
c, addr = s.accept()
print ('Got connection from', addr )

# send a thank you message to the client. encoding to send byte type.
c.send('Thank you for connecting'.encode())
while True:
if GPIO.input(7):
c.send("1".encode())
print("INtruder DEtected")
else:
c.send("0".encode())
print("No INtruder")
```

11. Write an Arduino sketch to blink an LED Light for a particular interval of time using wireless communication protocol(LoRa).

Transmitter Code:

```
#include <SPI.h>
#include <LoRa.h>
int pot = A0;
  void setup() {
  Serial.begin(9600);
  pinMode(pot,INPUT);

  while (!Serial);
  Serial.println("LoRa Sender");
  if (!LoRa.begin(433E6)) { // or 915E6, the MHz speed of yout module
    Serial.println("Starting LoRa failed!");
    while (1);
  }
}

void loop() {
  int val = map(analogRead(pot),0,1024,0,255);
  LoRa.beginPacket();
  LoRa.print(val);
```

```
  LoRa.endPacket();
  delay(50);

}
```

Receiver Code:

```
#include <SPI.h>
#include <LoRa.h>
int LED = 3;
String inString = "";    // string to hold input
int val = 0;

void setup() {
  Serial.begin(9600);
  pinMode(LED,OUTPUT);

  while (!Serial);
  Serial.println("LoRa Receiver");
  if (!LoRa.begin(433E6)) { // or 915E6
    Serial.println("Starting LoRa failed!");
    while (1);
  }
}
void loop() {
  // try to parse packet
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // read packet
    while (LoRa.available())
    {
      int inChar = LoRa.read();
      inString += (char)inChar;
      val = inString.toInt();
    }
    inString = "";
    LoRa.packetRssi();
  }

  Serial.println(val);
  analogWrite(LED, val);
}
```

Case study:

Assume that you are in a college, design and implement a IoT prototype to measure the amount of usage of water at a given location (take the location from user) on a day-to-day basis and send the information to Cloud.

Receive the above information from the sensors/ cloud and apply necessary algorithms to predict the amount of water being wasted at a particular location and also send a notification to the user.

Implement: The students has to implement based on their perceptions.