ECE2002

AEC PROJECT REPORT

Help fighting global pandemics like COVID -19

COVID-19 ROBOT

Under the Guidance of: Prof. Aarthy G

Submitted by

- **ADITYA JAIN-19BEC0122**
- **ANSHUMAN PHADKE -19BEC0428**
- **DASARI LOLITHA MANYA-19BEC0182**

The problem statement:

One of the most effective ways to prevent the spread of COVID-19 and other viral and cold diseases is to keep other people away. From my own experience I can confirm that this is not so easy outside of my own home. The best example is standing in line at the cash register or at a counter. A minimum distance of 1.5m (or more) should be maintained here. But who is walking around with a yardstick in front of them?

Our project proposal

A covid-19 robot to measure temperature and automatic floor disinfectant and ultrasonic distance warning tool

Recently wireless sensors and sensor networks plays a vital role in the research, technological community. But there are different from traditional wireless networks as well as computer networks, today the progress in science and technology offers miniature, speed, intelligence, sophistication, and new materials at lower cost, resulting in the development of various high-performance smart sensingsystem. At a time when the whole world is fighting the same enemy, we have all had to embrace new technologies and discover their benefits.
As our topic name suggests this is a robotic car which can be of good help where there are lots of people. This will be controlled using

1. L239D Motor Driver
2. HC-05 Bluetooth Module
3. Arduino UNO Microcontroller
4. PIR Motion Sensor
5. Temperature Sensor-DHT11
6. Arduino IDE for programming
7. Arduino Bluetooth Control Android App for Communication with Bluetooth Module

This Robot will regularly check the temperature and if a person has high temperature the data can be sent through the Bluetooth app. This paper proposes a perspective model which can be used to predict the susceptible population for COVID-19 test at early stage on the base of symptoms. It will also helpful to reduce the human intervention with the COVID -19 patients at early stage with the help of IoT.

As a result, COVID-19 may well have been the embodiment of the Internet of Things (IoT).We are facing a problem with only solution that appears right now is social distancing but also to treat patients and to contain the spread we have to separate the carriers from general population and at such aa time where distancing between medical staff and people who are carriers is a must technology comes at rescue and keeping in mind this problem we have come across an all in one robot that can at the same time sense people body temperature, help to sanitize their hands, and also as social distancing is must it can help people maintain a particular social gap. Further implementations can be using a Camera Module, Raspberry Pi, etc.

***Pseudo Code*:**

```
#define pirPin 10 //Defines the PIR Motion Sensor Pin
int red=8;//Red LED Pin
int green=9;//Green LED Pin
int calibrationTime = 30;
long unsigned int lowIn;
long unsigned int pause = 5000;
boolean lockLow = true;
boolean takeLowTime;
int PIRValue = 0;//Initializing the PIR Motion Sensor Value
#include <dht.h>
#define dht_apin A0 // DHT11 Sensor Pin
dht DHT;
int LeftMotorFront=2;
int LeftMotorBack=3;
int RightMotorFront=4;
int RightMotorBack=5;

int ln=0;//byte value
void setup() {
  // put your setup code here, to run once:
   Serial.begin(9600);//baud rate for communication with the serial monitor
   pinMode(pirPin, INPUT);
   pinMode(red,OUTPUT);
   pinMode(green,OUTPUT);
```

```
 pinMode(LeftMotorFront,OUTPUT);
 pinMode(LeftMotorBack,OUTPUT);
 pinMode(RightMotorFront,OUTPUT);
 pinMode(RightMotorBack,OUTPUT);
}

void loop() {
 // put your main code here, to run repeatedly:
 if(Serial.available()>0)//Only if there is some input from the serial monitor
{
  ln=Serial.read();
  Serial.println(ln);
  Serial.print("\n");
  if (ln==49)//The robot will move in the forward direction
 {
   digitalWrite(LeftMotorFront,HIGH);
   digitalWrite(RightMotorFront,HIGH);
   digitalWrite(LeftMotorBack,LOW);
   digitalWrite(RightMotorBack,LOW);
 }
 else if(ln==51)//The robot will move in the backward direction
 {
   digitalWrite(LeftMotorFront,LOW);
   digitalWrite(RightMotorFront,LOW);
   digitalWrite(LeftMotorBack,HIGH);
   digitalWrite(RightMotorBack,HIGH);
 }
 else if(ln==50)//The robot will move in the left direction
 {
   digitalWrite(LeftMotorFront,HIGH);
   digitalWrite(RightMotorFront,LOW);
   digitalWrite(LeftMotorBack,LOW);
   digitalWrite(RightMotorBack,HIGH);
 }
 else if(ln==52)//The robot will move in the right direction
 {
   digitalWrite(LeftMotorFront,LOW);
   digitalWrite(RightMotorFront,HIGH);

    digitalWrite(LeftMotorBack,HIGH);
   digitalWrite(RightMotorBack,LOW);
 }
}
```

```
sensor2();//The robot will move only on the basis of PIR motion sensor
}
//We define a function for DHT 11 Sensor
void sense1(){
    DHT.read11(dht_apin);
    int h=int(DHT.temperature);
    Serial.print("The body temperature is:");
    Serial.println(h);
}
//We define a function for PIR Motion Sensor
void sensor2(){
 if(digitalRead(pirPin) == HIGH) {
    if(lockLow) {
      PIRValue = 1;
      lockLow = false;
      Serial.println("Motion detected.");
      digitalWrite(green,HIGH);//Green LED will be  ON
      digitalWrite(red,LOW);//Red LED will be OFF
      sense1();//We call our function only when motion is detected,this will detect the temperature of the
human .
      delay(50);
    }
    takeLowTime = true;
  }
  if(digitalRead(pirPin) == LOW) {
    if(takeLowTime){
      lowIn = millis();takeLowTime = false;
    }
    if(!lockLow && millis() - lowIn > pause) {
      PIRValue = 0;
      lockLow = true;
      Serial.println("Motion ended.");
      digitalWrite(red,HIGH);//Red LED will be ON
      digitalWrite(green,LOW);//Green LED will be OFF
      delay(50);
    }//If no motion is detected then we will not call the dht sensor function because no motion is detected
  }
```

```
#define pirPin 10 //Defines the PIR Motion Sensor Pin
int red=8;//Red LED Pin
int green=9;//Green LED Pin
int calibrationTime = 30;
long unsigned int lowIn;
long unsigned int pause = 5000;
boolean lockLow = true;
boolean takeLowTime;
int PIRValue = 0;//Initializing the PIR Motion Sensor Value
#include <dht.h>
#define dht_apin A0 // DHT11 Sensor Pin
dht DHT;
int LeftMotorFront=2;
int LeftMotorBack=3;
int RightMotorFront=4;
int RightMotorBack=5;

int ln=0;//byte value
```

```
void setup() {
  // put your setup code here, to run once:
   Serial.begin(9600);//baud rate for communication with the serial moniter
   pinMode(pirPin, INPUT);
   pinMode(red,OUTPUT);
   pinMode(green,OUTPUT);

  pinMode(LeftMotorFront,OUTPUT);
  pinMode(LeftMotorBack,OUTPUT);
  pinMode(RightMotorFront,OUTPUT);
  pinMode(RightMotorBack,OUTPUT);
 }
```

```
void loop() {
  // put your main code here, to run repeatedly:
  if(Serial.available()>0)//Only if there is some input from the serial moniter
{
   ln=Serial.read();
   Serial.println(ln);
   Serial.print("\n");
   if (ln==49)//The robot will move in the forward direction
  {
    digitalWrite(LeftMotorFront,HIGH);
    digitalWrite(RightMotorFront,HIGH);
    digitalWrite(LeftMotorBack,LOW);
    digitalWrite(RightMotorBack,LOW);
  }
  else if(ln==51)//The robot will move in the backward direction
  {
    digitalWrite(LeftMotorFront,LOW);
    digitalWrite(RightMotorFront,LOW);
    digitalWrite(LeftMotorBack,HIGH);
    digitalWrite(RightMotorBack,HIGH);
  }
  else if(ln==50)//The robot will move in the left direction
  {
    digitalWrite(LeftMotorFront,HIGH);
    digitalWrite(RightMotorFront,LOW);
    digitalWrite(LeftMotorBack,LOW);
    digitalWrite(RightMotorBack,HIGH);
  }
  else if(ln==52)//The robot will move in the right direction
  {
    digitalWrite(LeftMotorFront,LOW);
    digitalWrite(RightMotorFront,HIGH);
    digitalWrite(LeftMotorBack,HIGH);
    digitalWrite(RightMotorBack,LOW);
  }
}
sensor2();//The robot will move only on the basis of PIR motion sensor
}
```

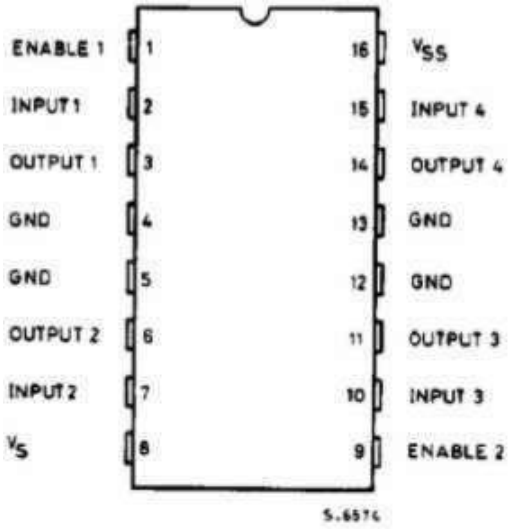
```
//We define a function for DHT 11 Sensor
void sensel(){
      DHT.readl1(dht_apin);
      int h=int(DHT.temperature);
      Serial.print("The body temperature is:");
      Serial.println(h);
}
//We define a function for PIR Motion Sensor
void sensor2(){
  if(digitalRead(pirPin) == HIGH) {
      if(lockLow) {
         PIRValue = 1;
         lockLow = false;
         Serial.println("Motion detected.");
         digitalWrite(green,HIGH);//Green LED will be  ON
         digitalWrite(red,LOW);//Red LED will be OFF
         sensel();//We call our function only when motion is detected,this will detect the temperature of the huma  .
         delay(50);
      }
      takeLowTime = true;
   }
   if(digitalRead(pirPin) == LOW) {
      if(takeLowTime){
         lowIn = millis();takeLowTime = false;
      }
      if(!lockLow && millis() - lowIn > pause) {
         PIRValue = 0;
         lockLow = true;
         Serial.println("Motion ended.");
         digitalWrite(red,HIGH);//Red LED will be ON
         digitalWrite(green,LOW);//Green LED will be OFF
         delay(50);
      }//If no motion is detected then we will not call the dht sensor function because no motion is detected.
   }
}
```

Using L239D Motor Driver

From microcontroller we cannot connect a motor directly because microcontroller cannot give sufficient current to drive the DC motors.

Motor driver is a current enhancing device, it can also be act as Switching Device. Thus, we insert motor driver in between motor and microcontroller. Motor driver take the input signals from
microcontroller and generate corresponding output for motor. **Motor Driver IC L293D**

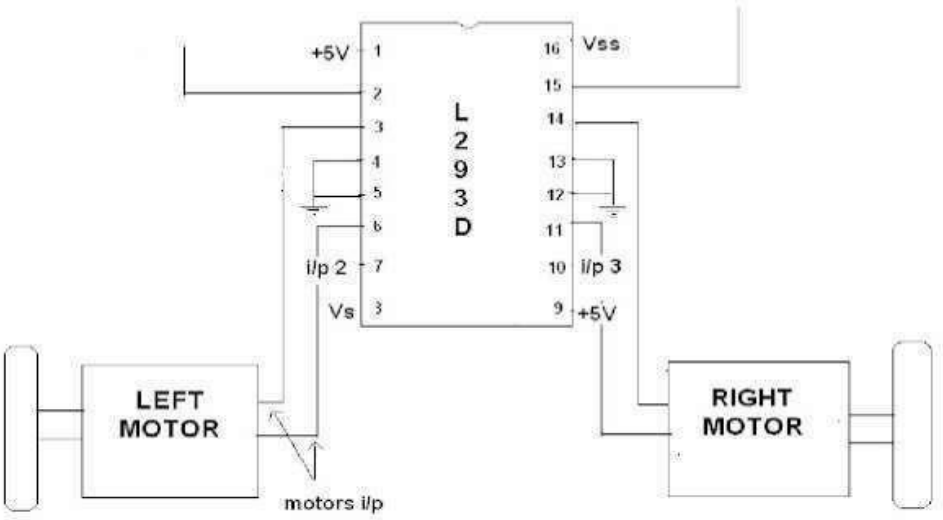
This is a motor driver IC that can drive two motor simultaneously. L293D IC is a dual H-bridge motor driver IC. One H-bridge is capable to drive a dc motor in bidirectional. L293D IC is a current enhancing IC as the output from the sensor is not able to drive motors itself so L293D is used for this purpose. L293D is a 16 pin IC having two enables pins which should always be remain high to enable both the H-bridges. L293B is another IC of L293 series having



1. L293D can run a motor up to 600 mA whereas L293B can run up to 1 A.

2. L293D has protection diode whereas L293B doesn't have any such protection diode. Need to
add the protection diode manually.

Connecting L239D to motors



Points regarding L293D

Supply voltage (Vss) is the Voltage at which we wish to drive the motor.
Generally, we prefer 6V for dc motor and 6 to 12V for gear motor, depending upon the rating of the motor.
Logical Supply Voltage will decide what value of input voltage should be considered as high or low. So, if we set Logical Supply Voltage equals to +5V, then - 0.3V to 1.5V will be considered as Input Low Voltage and 2.3 V to 5 V will be considered as Input High Voltage.
L293D has 2 Channels. *One channel is used for one motor*.

- Channel 1 - Pin 1to 8
- Channel 2 - Pin 9 to 16

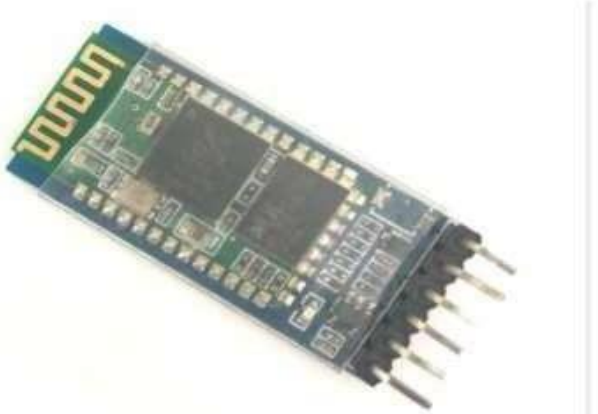
Enable Pin is use to enable or to make a channel active. Enable pin is also called as Chip Inhibit Pin.
All Input (Pin No. 2, 7,10and 15) of L293D IC is the output from microcontroller (*ATmega8*)

if Enable pin low, the output will be at 0 always. If its high output depends on input

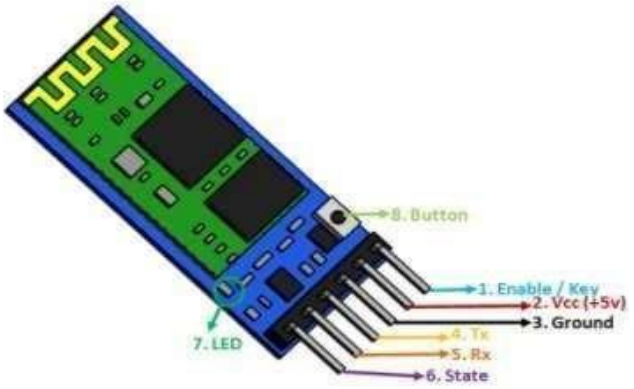
Output Connections

- OUTPUT 1 (Pin No 3) --- Negative Terminal of Right Motor
- OUTPUT 2 (Pin No 6) --- Positive Terminal of Right Motor
- OUTPUT 3 (Pin No 10) --- Positive Terminal of Left Motor
- OUTPUT 4 (Pin No 14) --- Negative Terminal of Left Motor

Using Bluetooth Module HC 05



HC-05 Bluetooth Module



HC-05 Bluetooth Module Pinout

HC-05 Technical Specifications

- Serial Bluetooth module for Arduino and other microcontrollers
- Operating Voltage: 4V to 6V (Typically +5V)
- Operating Current: 30mA
- Range: <100m
- Works with Serial communication (USART) and TTL compatible
- Follows IEEE 802.15.1 standardized protocol
- Uses Frequency-Hopping Spread spectrum (FHSS)
- Can operate in Master, Slave or Master/Slave mode
- Can be easily interfaced with Laptop or Mobile phones with Bluetooth
- Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.

Where to use HC-05 Bluetooth module

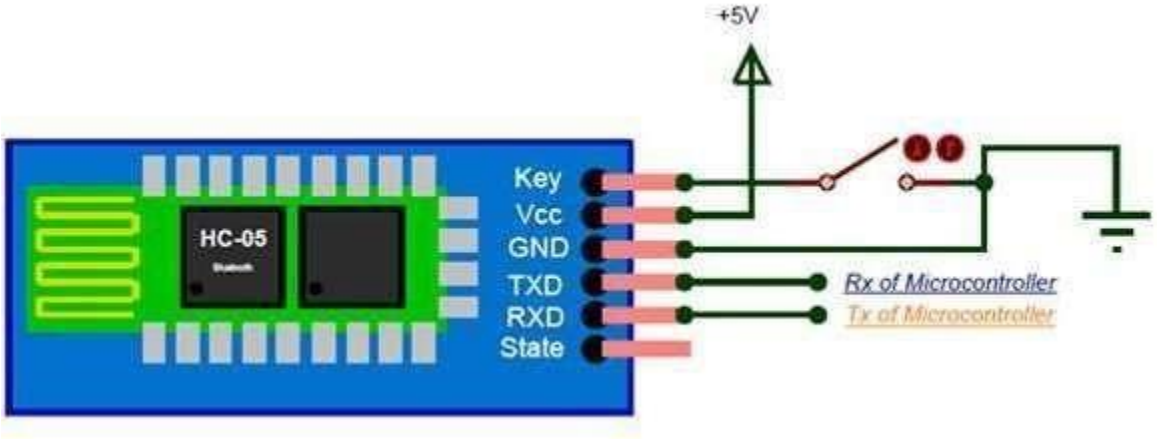
The **HC-05** is a very cool module which can add two-way (full-duplex) wireless functionality to your projects. You can use this module to communicate between two microcontrollers like Arduino or communicate with any device with Bluetooth functionality like a Phone or Laptop. There are many android applications that are already available which makes this process a lot easier. The module communicates with the help of USART at 9600 baud rate hence it is easy to interface with any microcontroller that supports USART. We can also configure the default values of the module by using the command mode. So, if you looking for a Wireless module that could transfer data from your computer or mobile phone to microcontroller or vice versa then this module might be the right choice for you. However, do not expect this module to transfer multimedia like photos or songs; you might have to look into the CSR8645 module for that.

How to Use the HC-05 Bluetooth module

The **HC-05** has two operating modes, one is the Data mode in which it can send and receive data from other Bluetooth devices and the other is the AT Command mode where the default device settings can be

changed. We can operate the device in either of these two modes by using the key pin as explained in the pin description.

It is very easy to pair the HC-05 module with microcontrollers because it operates using the Serial Port Protocol (SPP). Simply power the module with +5V and connect the Rx pin of the module to the Tx of MCU and Tx pin of module to Rx of MCU as shown in the figure below

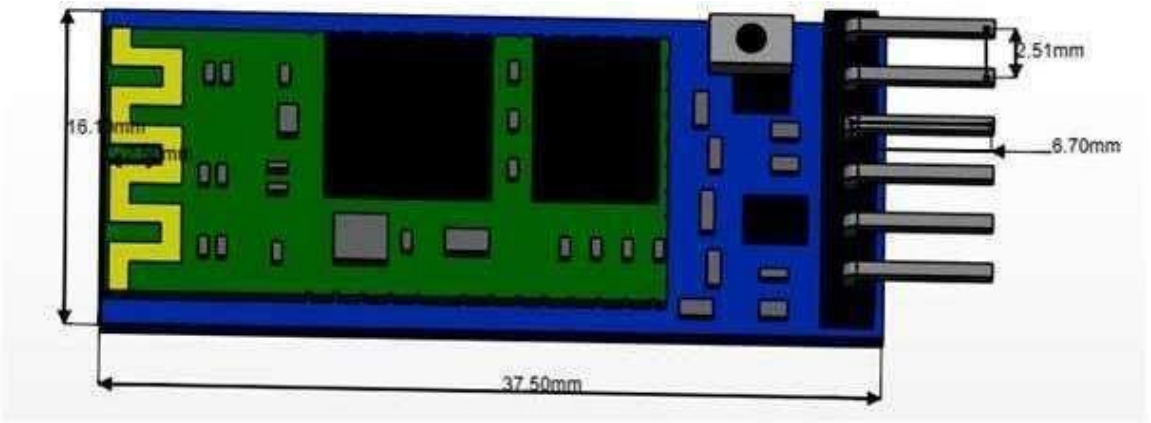


During power up the key pin can be grounded to enter into Command mode, if left free it will by default enter into the data mode. As soon as the module is powered you should be able to discover the Bluetooth device as "HC-05" then connect with it using the default password 1234 and start communicating with it. The name password and other default parameters can be changed by entering into the

Applications

1. Wireless communication between two microcontrollers
2. Communicate with Laptop, Desktops and mobile phones
3. Data Logging application
4. Consumer applications
5. Wireless Robots
6. Home Automation

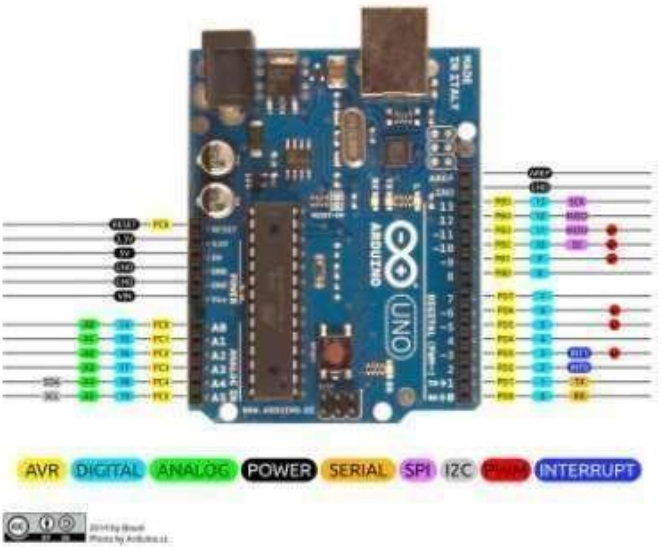
2D Model



Pin Configuration

Pin Number	Pin Name	Description
1	Enable / Key	This pin is used to toggle between Data Mode (set low) and AT command mode (set high). By default, it is in Data mode
2	Vcc	Powers the module. Connect to +5V Supply voltage
3	Ground	Ground pin of module, connect to system ground.
4	TX – Transmitter	Transmits Serial Data. Everything received via Bluetooth will be given out by this pin as serial data.
5	RX – Receiver	Receive Serial Data. Every serial data given to this pin will be broadcasted via Bluetooth
6	State	The state pin is connected to on board LED, it can be used as a feedback to check if Bluetooth is working properly.
7	LED	Indicates the status of Module • Blink once in 2 sec: Module has entered CommandMode • Repeated Blinking: Waiting for connection in Data Mode • Blink twice in 1 sec: Connection successful in Data Mode
8	Button	Used to control the Key/Enable pin to toggle between Data and command Mode

Arduino Uno



Arduino Uno Pin Diagram

Arduino Uno is a microcontroller board based on 8-bit ATmega328P microcontroller. Along with ATmega328P, it consists other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller. Arduino Uno has 14 digital input/output pins (out of which 6 can be used as PWM outputs), 6 analog input pins, a USB connection, A Power barrel jack, an ICSP header and a reset button.

How to use Arduino Board

The 14 digital input/output pins can be used as input or output pins by using pinMode(), digitalRead() and digitalWrite() functions in arduino programming. Each pin operate at 5V and can provide or receive a maximum of 40mA current, and has an internal pull-up resistor of 20-50 KOhms which are disconnected by default. Out of these 14 pins, some pins have specific functions as listedbelow:

Serial Pins 0 (Rx) and 1 (Tx): Rx and Tx pins are used to receive and transmit TTL serial data. They are connected with the corresponding ATmega328P USB to TTL serialchip.

External Interrupt Pins 2 and 3: These pins can be configured totrigger an interrupt on alow value, a rising or falling edge, or a change invalue.

PWM Pins 3, 5, 6, 9 and 11: These pins provide an 8-bit PWM output by using analogWrite() function.

SPI Pins 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK): These pins are used for SPI communication.

In-built LED Pin 13: This pin is connected with an built-in LED, when pin 13 is HIGH – LED is on and when pin 13 is LOW, its off.

Along with 14 Digital pins, there are 6 analog input pins, each of which provide 10 bits of resolution, i.e. 1024 different values. They measure from 0 to 5 volts but this limit can be increased by using AREF pin with analog Reference () function.

Analog pin 4 (SDA) and pin 5 (SCA) also used for TWI communication using Wire library.

Arduino Uno has a couple of other pins as explained below:

AREF: Used to provide reference voltage for analog inputs with analogReference() function.

Reset Pin: Making this pin LOW, resets the microcontroller.

Pin Description

Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on theboard. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

Arduino Uno Technical Specifications

Microcontroller	ATmega328P – 8 bit AVR family microcontroller
Operating Voltage	5V
Recommended Input Voltage	7-12V
Input Voltage Limits	6-20V
Analog Input Pins	6 (A0 – A5)
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC Current on I/O Pins	40 mA
DC Current on 3.3V Pin	50 mA
Flash Memory	32 KB (0.5 KB is used for Bootloader)
SRAM	2 KB
EEPROM	1 KB
Frequency (Clock Speed)	16 MHz

Arduino Uno to ATmega328 Pin Mapping

When ATmega328 chip is used in place of Arduino Uno, or vice versa, the image below shows the pin mapping between the two.

Arduino function reset	TPCINT14/RESET) PC6E	.	2	PC5(AOC5/SCL/PCINT13)	Ardulno function analog›npul 5
digital p‹n0 (PXI	(PCINT1g/RXD) PDOE a	zt	2	<I DC4/SDA/PCINT12)	ai\alog ir1put 4
digital pin 1 (TK)	{PCINT17/TXD) PDF 3	zs	**]	PCS (AOC3/PCINT11)	analog inpul 8
digital pin 2	\PCINT18/INTOj PD2E ‹		2	PC2(AOC2/PCINT1O\	analog inputP
digital pin 3 (PWM)	(PCINT19/DG2B/INT1) Ph E •	•‹	Z	1{ADC /PCINT9›	analog input1
rligilal pin 4	(PCINT2O/XCK/TO) PD4L •		2	PCOtADC0/PCINT8)	analr›g input 0
VCC	VCCU•	u		GND	GND
GND	GNDL e	z‹	2	AREP	analcg reference
cryslal	(PCINT8/XTAL1/TOSC1)PB8/ e	a	Z	AVCC	VCC
crysial	(PCINT7/XTAL2/TOSC2) PB7/ 0		4**	]PBX(SCK/PClhIT5)	dl0›lal p•n I3
digital pin 5 (PWM)	\PCINT21/OCOB/T1) PDSE ‹	‹s	2	PB4 (MISO/PClhIT4)	cigilal p›n T2
oig›lal p•n G (PWM)	{PCINT23/OCOA/AINO) P 12	17		PB3 (MOSI/OC8A/PCINT3	digital pin 11\PwM}
digital p n 7	(PCJNT23/AIN1) PD7/ s	‹s		PB2 (SS/QC1B/PCINT2)	dirjital pin 10 (PWM]
digital pin 8	(PC fNTO/CLKOziCP1} P 14	1?		PBX (1A/PCINT1)	dxgilal piri g (PWM)

Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header

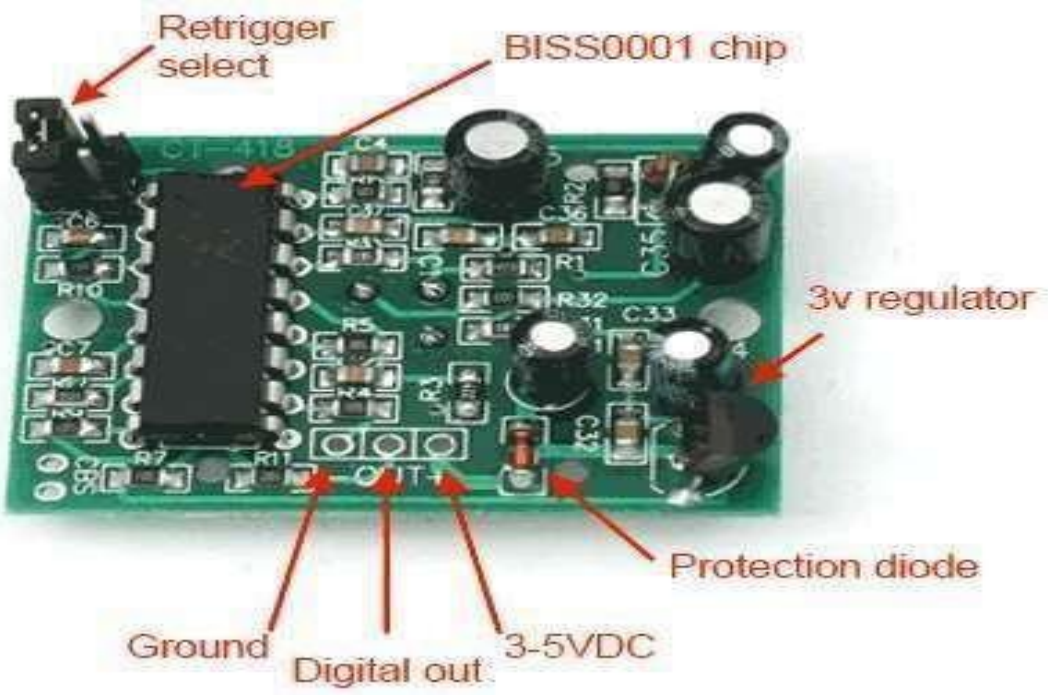
PIR sensor

PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensors range. They are small, inexpensive, low-power, easy to use and don't wear out. For that reason they are commonly found in appliances and gadgets used in homes or businesses. They are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors.

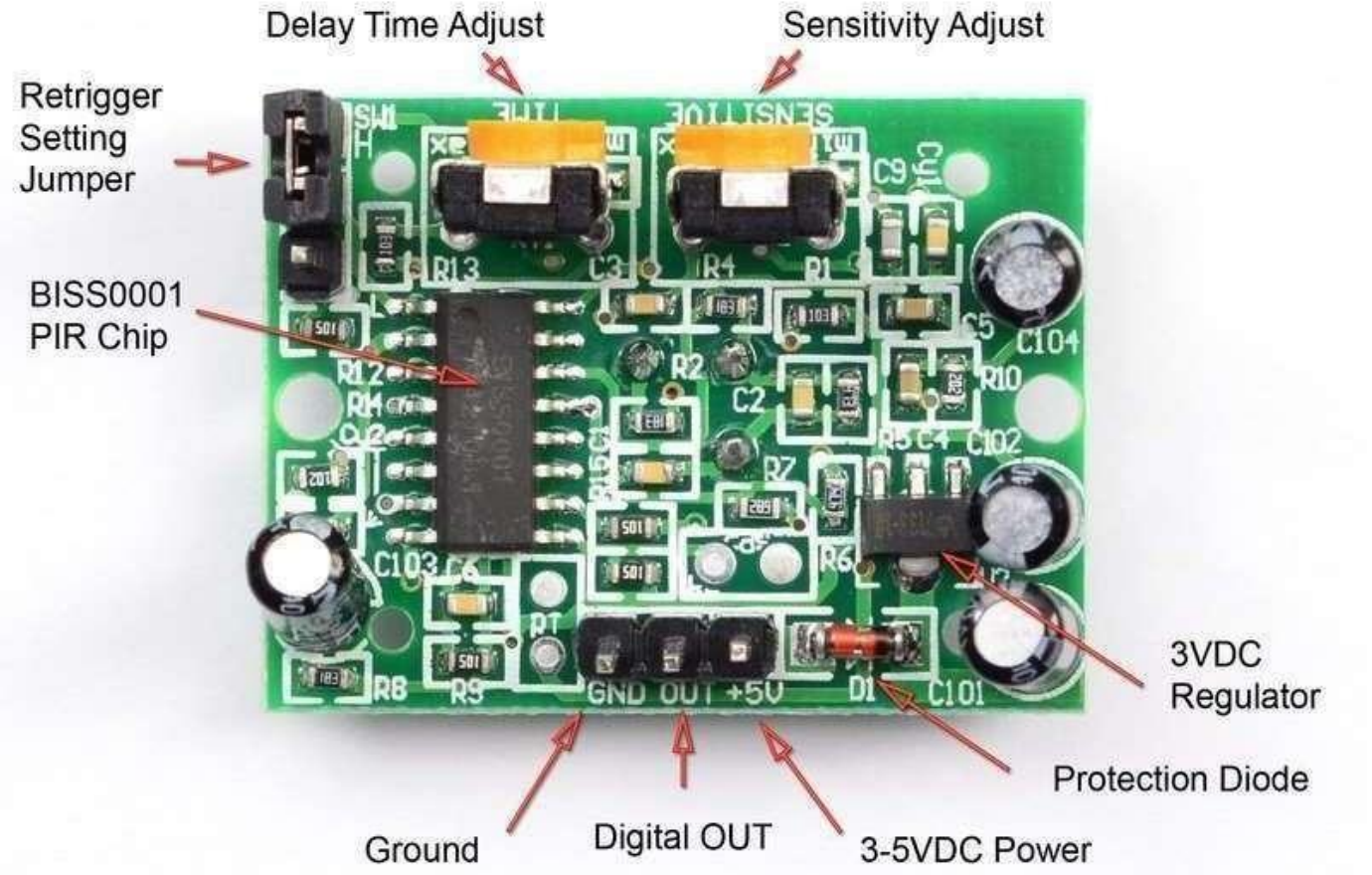
PIRs are basically made of a pyroelectric sensor (which you can see below as the round metal can with a rectangular crystal in the center), which can detect levels of infrared radiation. Everything emits some low level radiation, and the hotter something is, the more radiation is emitted. The sensor in a motion detector is actually split in two halves. The reason for that is that we are looking to detect motion (change) not average IR levels. The two halves are wired up so that they cancel each other out. If one half sees more or less IR radiation than theother, the output will swing high or low.

Along with the pyroelectic sensor is a bunch of supporting circuitry, resistors and capacitors. It seems that most small hobbyist sensors use the BISS0001 ("Micro Power PIR Motion Detector IC"), undoubtedly a very inexpensive chip. This chip takes the output of the sensor and does some minor processing on it to emit a digital output pulse from the analog sensor.

Our older PIRs looked like this:



Our new PIRs have more adjustable settings and have a header installed in the 3 -pin ground/out/power pads



For many basic projects or products that need to detect when a person has left or entered the area, or has approached, PIR sensors are great. They are low power andlow cost, pretty rugged, have a wide lens range, and are easy to interface with. Note that PIRs won't tell you how many people are around or how close they are to the sensor, the lens is often fixed to a certain sweep and distance (although it can be hacked somewhere) and they are also sometimes set off by housepets. Experimentation is key!

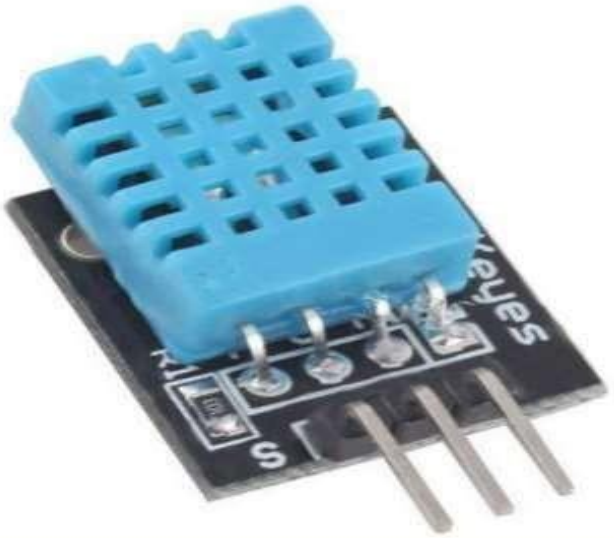
Some Basic Stats

These stats are for the PIR sensor in the Adafruit shop which is very much like the Parallax one . Nearly all PIRs will have slightly different specifications, although they all pretty much work the same. If there's a datasheet, you'll want to refer to it

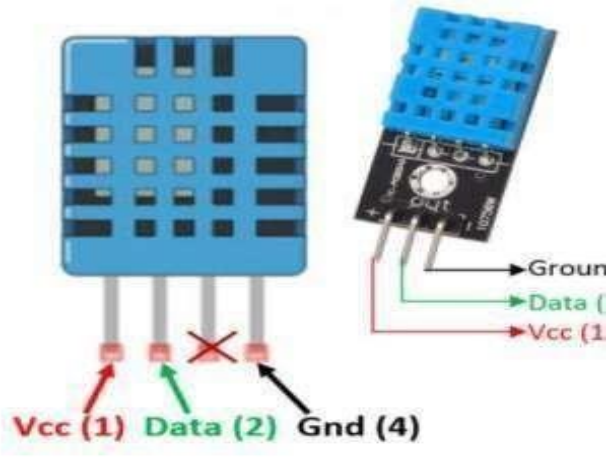
- **Size:** Rectangular
- **Price:** $10.00 at the Adafruit shop
- **Output:** Digital pulse high (3V) when triggered (motion detected) digital low when idle (no motion detected). Pulse lengths are determined by resistors and capacitors on the PCB and differ from sensor to sensor.
- **Sensitivity range:** up to 20 feet (6 meters) 110° x 70° detection range
- **Power supply:** 5V-12V input voltage for most modules (they have a 3.3V regulator), but 5V is ideal in case the regulator has different specs

- **BIS0001 Datasheet** (the decoder chip used)
- **RE200B datasheet** (most likely the PIR sensing element used)
- **NL11NH datasheet** (equivalent lens used)
- **Parallax Datasheet on their version of the sensor**

Temperature Sensor



DHT11–Temperature and Humidity Sensor



DHT11 Sensor Pinout

Pin Identification and Configuration:

No:	Pin Name	Description
For DHT11 Sensor		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	NC	No Connection and hence not used
4	Ground	Connected to the ground of the circuit
For DHT11 Sensor module		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	Ground	Connected to the ground of the circuit

DHT11 Specifications:

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

Difference between DHT11 Sensor and module:

The **DHT11 sensor** can either be purchased as a sensor or as a module. Either way, the performance of the sensor is same. The sensor will come as a 4-pin package out of which only three pins will be used whereas the module will come with three pins as shown above.

The only difference between the sensor and module is that the module will have a filtering capacitor and pull-up resistor inbuilt, and for the sensor, you have to use them externally if required.

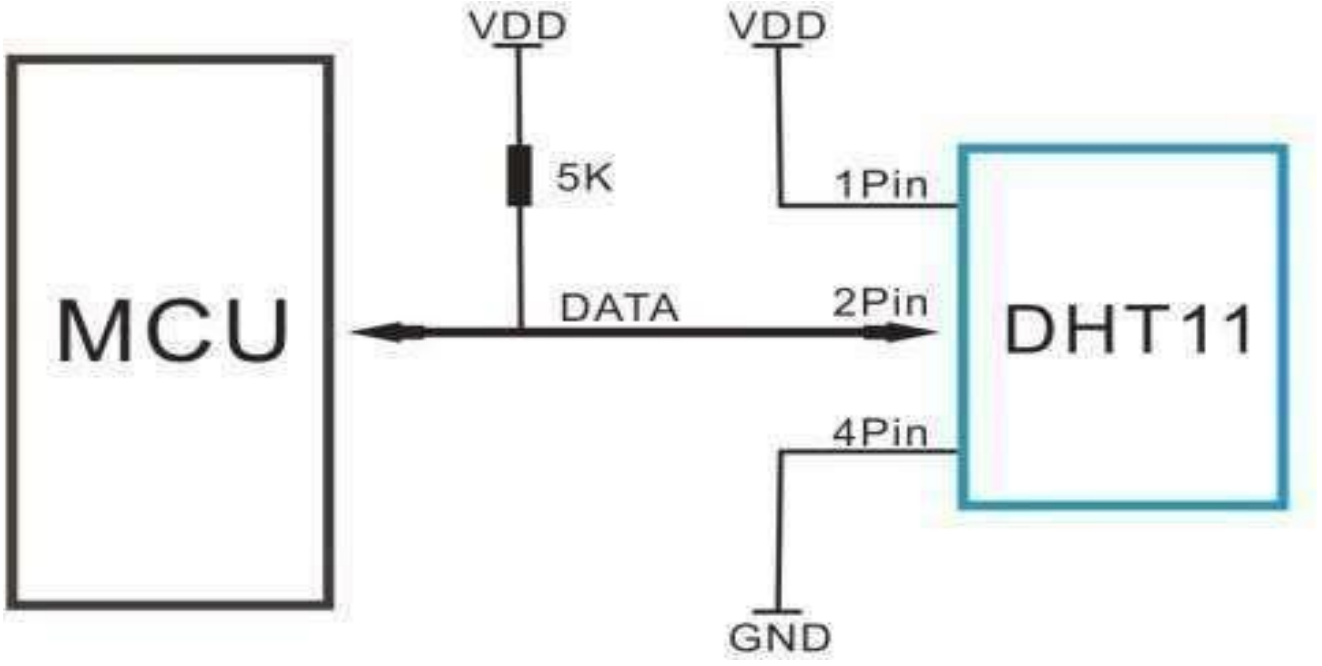
Where to use DHT11:

The **DHT11** is a commonly used **Temperature and humidity sensor.** The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. The sensor is also factory calibrated and hence easy to interface with other microcontrollers.

The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%. So if you are looking to measure in this range then this sensor might be the right choice for you.

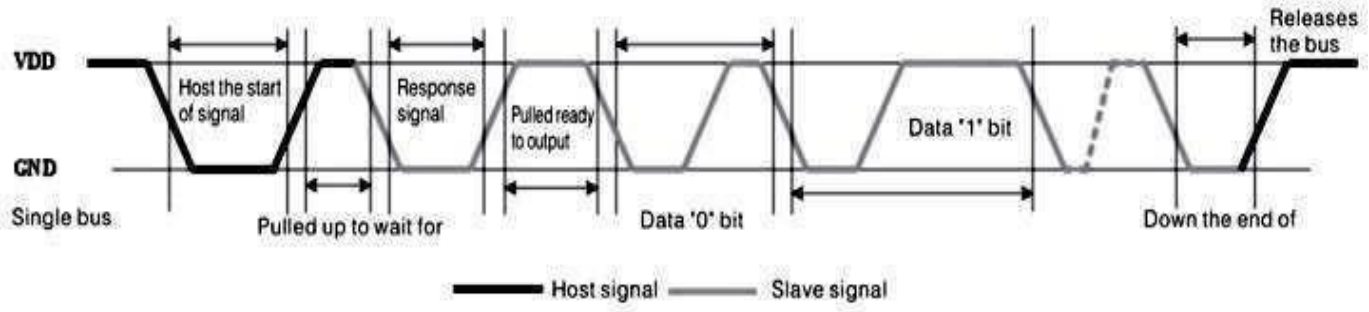
How to use DHT11 Sensor:

The DHT11 Sensor is factory calibrated and outputs serial data and hence it is highly easy to set it up. The connection diagram for this sensor is shown below.



As you can see the data pin is connected to an I/O pin of the MCU and a 5K pull-up resistor is used. This data pin outputs the value of both temperature and humidity as serial data. If you are trying to interface DHT11 with Arduino then there are ready-made libraries for it which will give you a quick start.

If you are trying to interface it with some other MCU then the datasheet given below will come in handy. The output given out by the data pin will be in the order of 8bit humidity integer data + 8bit the Humidity decimal data +8 bit temperature integer data + 8bit fractional temperature data +8 bit parity bit. To request the DHT11 module to send these data the I/O pin has to be momentarily made low and then held high as shown in the timing diagram below

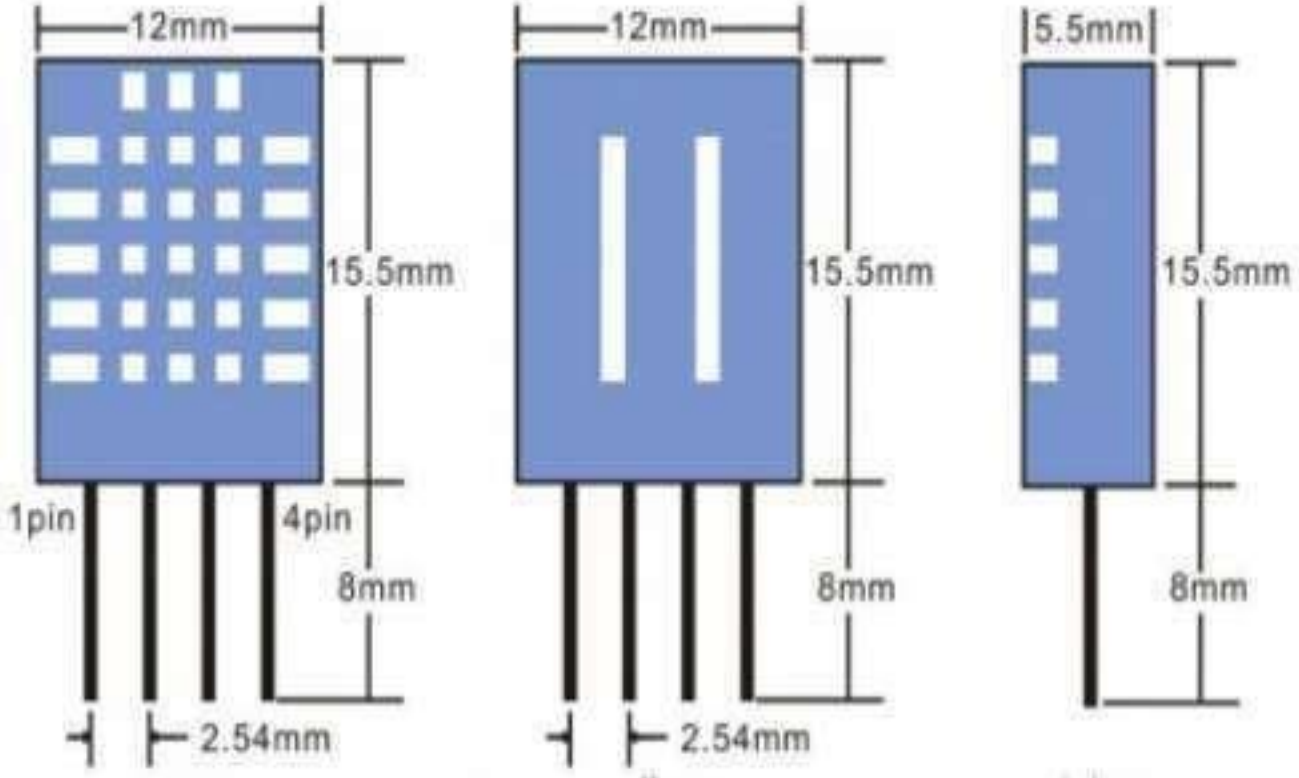


The duration of each host signal is explained in the DHT11 datasheet, with neat steps and illustrative timing diagrams

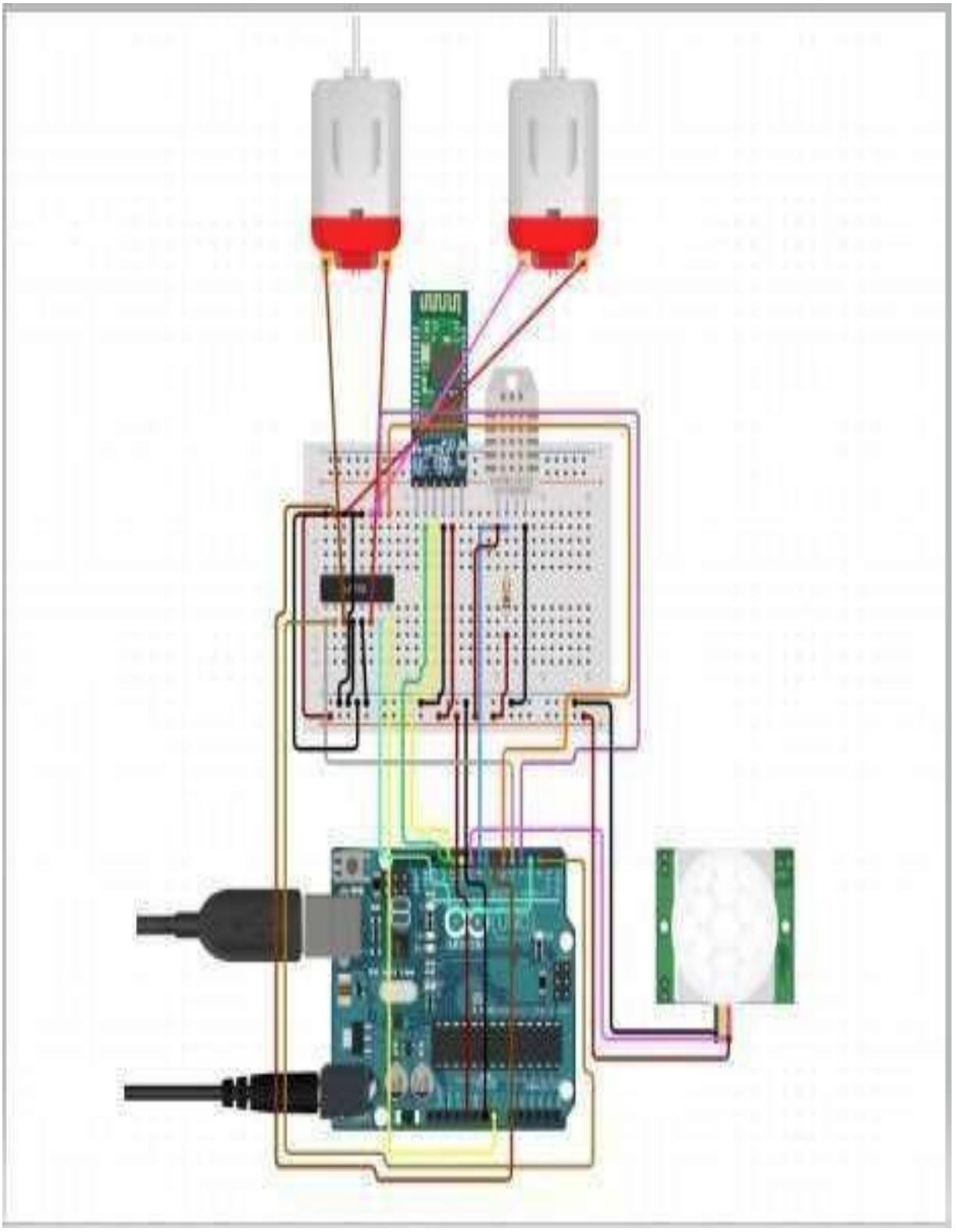
Applications:

- Measure temperature and humidity
- Local Weather station
- Automatic climate control
- Environment monitoring

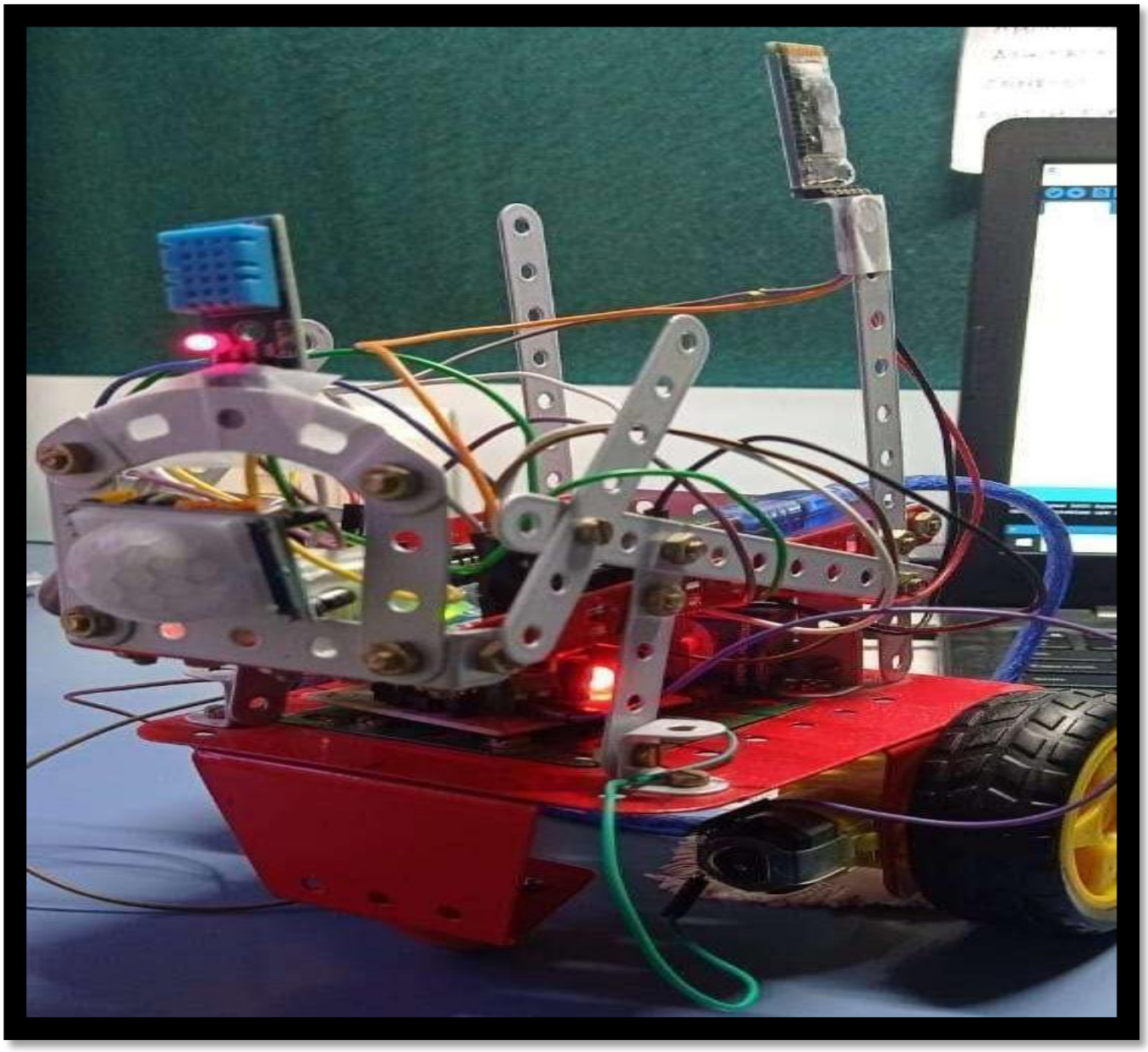
2D–model of the sensor:



Implementation steps: basic connections online simulation



OUR PROTOTYPE



Conclusion And Further Implementation:

In this work, we have proposed a unique model for future **IoT-based healthcare systems**, which can be applied to both general systems and systems that monitor specific conditions. We then presented a thorough and systematic overview of the state-of-the-art works relating to each component of the proposed model.

Recent works utilizing **cloud technologies** for data storage were presented, and showed that cloud is the best means for storing and organizing big data in healthcare. It is also shown by several works that significantly better data processing can be performed in the cloud than can be performed by wearable devices with their limited resources. The most significant drawback of using cloud is that it introduces security risks, and as such we presented several works focused on improving security in the cloud.

It was found that access control policies and encryption can significantly enhance security, but that no known standard is suitable for immediate application into a wearable, IoT-based healthcare system. Based on our analysis of state-of-the-art technologies in the fields of wearable sensors, communications standards, and cloud technology, we identified several significant areas for future research.

Machine learning and the development of a secure yet lightweight encryption scheme for cloud storage were the two areas that provide the most opportunity for researchers seeking to make significant improvements in the field of IoT-based healthcare.

References:

[1] Australian Institute of Health and Welfare, "Australia's Health," 2014. [Online]. Available: http://www.aihw.gov.au/WorkArea/DownloadAsset.aspx- ?id=60129548150

[2] E. Perrier, Positive Disruption: Healthcare, Ageing & Participation in the Age of Technology. Australia: The McKell Institute, 2015.

[3] P. Gope and T. Hwang, "BSN-Care: A Secure IoTBased Modern Healthcare System Using Body Sensor Network," IEEE Sensors Journal, vol. 16, no. 5, pp. 1368–1376, 2016.

[4] N. Zhu, T. Diethe, M. Camplani, L. Tao, A. Burrows, N. Twomey, D. Kaleshi, M. Mirmehdi, P. Flach, and I. Craddock, "Bridging e-Health and the Internet of Things: The SPHERE Project," IEEE Intelligent Systems, vol. 30, no. 4, pp. 39–46, 2015.

[5] S. H. Chang, R. D. Chiang, S. J. Wu, and W. T. Chang, "A Context-Aware, Interactive M-Health System for Diabetics," IT Professional, vol. 18, no. 3, pp. 14–22, 2016.

[6] C. F. Pasluosta, H. Gassner, J. Winkler, J. Klucken, and B. M. Eskofier, "An emerging era in the management of Parkinson's disease: Wearable technologies and the internet of things," IEEE Journal of Biomedical and Health Informatics, vol. 19, no. 6, pp. 1873– 1881, 2015.

[7] Y. J. Fan, Y. H. Yin, L. D. Xu, Y. Zeng, and F. Wu, "IoTbased smart rehabilitation system," IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pp. 1568–1577, 2014.

[8] S. Sarkar and S. Misra, "From Micro to Nano: The Evolution of Wireless Sensor- Based Health Care," IEEE Pulse, vol. 7, no. 1, pp. 21–25, 2016.

[9] Y. YIN, Y. Zeng, X. Chen, and Y. Fan, "The internet of things in healthcare: An overview," Journal of Industrial Information Integration, vol. 1, pp. 3–13, 3 2016.

[10] S. M. R. Islam, D. Kwak, H. Kabir, M. Hossain, and K.-S. Kwak, "The Internet of Things for Health Care : A Comprehensive Survey," IEEE Access, vol. 3, pp. 678 – 708, 2015.