PS02EMCA40 Embedded Systems and IoT

Dr. J. V. Smart

Table of Contents

Syllabus

COURSE NO: PS02EMCA40

Embedded Systems and IoT

w.e.f. June 2020

(3 Lectures & 1 Seminar/Tutorial per Week Total Marks: 100)

LEARNING OBJECTIVES

- To learn the fundamentals of embedded systems
- To understand the concepts, techniques, characteristics and applications of Internet of Things
- To gain an understanding of developing small/medium sized IoT projects using AVR, Arduino and other components
- To gain an understanding of developing IoT projects using the Raspberry Pi

PREREQUISITES

- Knowledge of computer programming
- Knowledge of the Python programming language

OUTCOMES OF THE COURSE

- Understanding of the fundamentals of embedded systems
- Knowledge of the definition, characteristics and applications of Internet of Things
- Familiarity with the hardware elements of IoT and the communication protocols commonly used with IoT
- Understanding of working with sensors, actuators and other devices
- Appreciation of security and privacy issues with IoT
- Basic knowledge of developing AVR/Arduino based IoT projects
- Basic knowledge of developing Raspberry Pi based IoT projects

COURSE CONTENT

1. **Introduction to Embedded Systems**
    - An introduction to embedded systems
    - Types and applications of embedded systems
    - The embedded system constraints: processing constraints, memory constraints, input/output constraints, response time constraints, predictability/reliability constraints
    - Processing units: microprocessors, microcontrollers, SoCs, ASICs, DSPs, FPGAs, etc.
    - Unique characteristics of embedded systems programming
2. **Introduction to Internet of Things**
    - Definition and characteristics of Internet of Things (IoT)
    - Applications of IoT in various domains
    - Hardware elements of IoT and their characteristics
    - Communication protocols commonly used with IoT
    - Sensors, actuators and other devices employed in IoT
    - Security and privacy concerns in IoT
3. **Development of Small/Medium Sized IoT Projects**
    - Introduction to AVR microcontollers
    - Introduction to the Arduino
    - Interfacing with the Arduino
    - Arduino shields
    - Arduino programming and the Arduino IDE
    - Wireless control and communications with the Arduino
4. **Development of IoT projects using the Raspberry Pi**
    - Introduction to the Raspberry Pi
    - Installing operating system and software on the Raspberry Pi
    - Interfacing with the Raspberry Pi
    - Raspberry Pi hats
    - Developing projects using the Raspberry Pi

MAIN REFERENCE BOOKS

1. Prasad, K. V. K. K.: Embedded / Real-Time Systems – Concepts, Design & Programming Black Book, New Edition, Dreamtech Press, 2009.
2. Bahga, A., Madisetti, V.: Internet of Things – A Hands-on Approach, Universities Press, 2014.
3. Hoile C., et al.: Make – Raspberry Pi and AVR Projects, MakerMedia, 2014.
4. Margolis, M.: Arduino Cookbook, O'Reilly, 2nd Edition, 2011.
5. Halfacree, G.: The Official Raspberry Pi Beginner's Guide, Raspberry Pi Press, 2018.

ADDITIONAL REFERENCES

1. Hughes, J. M.: Arduino – A Technical reference, O'Reilly (SPD), 2017.
2. Monk, S.: Raspberry Pi Cookbook, O'Reilly (SPD), 2014.
3. Richardson, M., Wallace, S.: Make – Getting Started with Raspberry Pi, 2nd Edition, MakerMedia, 2015.

WEB REFERENCES

1. Embedded Systems, Wikibook, https://en.wikibooks.org/wiki/Embedded_Systems.
2. The Official Raspberry Pi Beginner's Guide (online), https://www.raspberrypi.org/magpi-issues/Beginners_Guide_v1.pdf.
3. The Official Raspberry Pi Projects Book (online), https://www.raspberrypi.org/magpi-issues/Projects_Book_v1.pdf.

Glossary

Multimeter
a device used to measure various properties of electric and electronic devices

PCB
Printed Circuit Board

Microcontroller
a microcontroller or MCU (microcontroller unit) is a small computer on a single chip. It contains processor(s), primary / main memory and secondary memory (flash memory). It may also contain integrated ADC, DAC, peripheral device controllers, firmware, etc.

SoC
System on Chip. A complete computer with processor, storage, RAM, etc. on a single chip.

SBC
Single-board Computer. A full-fledged computer that comes in the form of a single printed circuit board. Examples include Arduino, NodeMCU and Raspberry Pi

ADC
Analog to Digital converter

DAC
Digital to Analog converter

Firmware
hardware (ROM / EEPROM) + Software embedded / stored in ROM

AVR
a family of microcontrollers originally developed by Atmel. Most Arduino boards have an AVR microcontroller

JTAG
A debugging interface for embedded systems

Sensor
A sensor is a device that *senses* or collects some type of information from the physical environment and provides it to the embedded system / IoT device as input in the form

of analog or digital electrical / electronic signals. Examples are light sensor, temperature sensor, distance sensor, accelerometers / gyroscopes (for detecting changes in motion), proximity sensor (nearness sensor) etc.

Actuator

An actuator is a device that takes the analog / digital signal output from an embedded system / IoT device and performs some action in the physical world or changes the physical world in some way. Examples are LEDs, electric motors, relays, robotic arms, etc.

POS (Point Of Sale) Terminal

This is a small hand-held device used at the point of sale for capturing data, accepting payment or printing invoice / bill / ticket. Examples are bus ticketing machine, card payment accepting machine, etc.

Power Pin

A pin that can supply power to a device (typically 3.3V or 5V DC, 5-20 mA)

Ground Pin

A Ground (GND) pin is *held to ground* i.e. 0V DC

GPIO Pin

A GPIO (General Purpose Input Output) pin can be used for performing input / output of electrical / electronic signals

PWM

Pulse Width Modulation

Introduction

Embedded Systems Wikibook

Embedded Systems

Applications of Embedded Systems

Embedded systems are commonly found in consumer, industrial, automotive, home appliances, medical, telecommunication, commercial and military applications.

Telecommunications systems employ numerous embedded systems from telephone switches for the network to cell phones at the end user. Computer networking uses dedicated routers and network bridges to route data.

Consumer electronics include MP3 players, television sets, mobile phones, video game consoles, digital cameras, GPS receivers, and printers. Household appliances, such as microwave ovens, washing machines and dishwashers, include embedded systems to provide flexibility, efficiency and features. Advanced HVAC systems use networked thermostats to more accurately and efficiently control temperature that can change by time of day and season. Home automation uses wired- and wireless-networking that can be used to control lights, climate, security, audio/visual, surveillance, etc., all of which use embedded devices for sensing and controlling.

Transportation systems from flight to automobiles increasingly use embedded systems. New airplanes contain advanced avionics such as inertial guidance systems and GPS receivers that also have considerable safety requirements. Various electric motors — brushless DC motors, induction motors and DC motors — use electronic motor controllers. Automobiles, electric vehicles, and hybrid vehicles increasingly use embedded systems to maximize efficiency and reduce pollution. Other automotive safety systems using embedded systems include anti-lock braking system (ABS), Electronic Stability Control (ESC/ESP), traction control (TCS) and automatic four-wheel drive.

Medical equipment uses embedded systems for monitoring, and various medical imaging (PET, SPECT, CT, and MRI) for non-invasive internal inspections. Embedded systems within medical equipment are often powered by industrial computers.

Embedded systems are used for safety-critical systems. Unless connected to wired or wireless networks via on-chip 3G cellular or other methods for IoT monitoring and control purposes, these systems can be isolated from hacking and thus be more secure. For fire safety, the systems can be designed to have a greater ability to handle higher temperatures and continue to operate. In dealing with security, the embedded systems can be self-sufficient and be able to deal with cut electrical and communication systems.

Miniature wireless devices called motes are networked wireless sensors. Wireless sensor networking makes use of miniaturization made possible by advanced IC design to couple full wireless subsystems to sophisticated sensors, enabling people and companies to measure a myriad of things in the physical world and act on this information through monitoring and control systems. These motes are completely self-contained and will typically run off a battery source for years before the batteries need to be changed or charged.

Vehicles

Engine (for fuel efficiency), braking system (ABS), power steering, power windows, automatic door lock, automatic transmission (auto gear shift)

Internet of Things (IoT)

Definition

IoT (Internet of Things)

> The Internet of things (IoT) describes physical objects (or groups of such objects) that are embedded with sensors, processing ability, software, and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks.

World's first IoT device was a modified Coca-Cola vending machine installed at the Carnegie Mellon University in 1982. It was able to report its inventory and whether newly loaded drinks were cold or not over the Internet.

IoT Device Hubs

Often, IoT devices are not powerful or smart enough to communicate with the Internet or the server directly. It may not be efficient also. In such situations, several IoT devices connect to the Internet or to the server through a hub.

Hardware Elements of IoT

An IoT ecosystem consists of several components

- **IoT devices** collect information and may take action also
- **IoT device hubs** helps several IoT devices connect to the Internet / could-based server
- **Network connectivity** IoT devices must have some form of network connectivity for communicating data. They may connect either directly to the Internet or use IoT device hubs for data transfer. In some cases, the IoT device or hub may store the data and then transfer in batches to improve efficiency and to reduce the load on the network
- **Cloud-based service** / **servers** collect the data from a large number of IoT devices and carry out back-end processing. They also send data / commands to the IoT devices as per requirement

Characteristics of IoT devices

IoT devices have the same characteristics as embedded systems. In addition, they have some form of network connectivity. The other hardware elements of the IoT ecosystems also often have typical characteristics

- Small size
- Low-cost

- Low processing power
- Very low amount of memory
- Very low amount of storage
- Restrictions on power consumption because of reliance on batteries for power
- Some IoT devices may have very limited or no user interface
- Huge variation in hardware
- Huge variation in software development tools
- Debugging is often very difficult
- Usually, it may be very difficult or impossible to patch or update the firmware / software once the device leaves the factory
- Network communication may be very slow and / or unreliable
- The procedure of connecting the IoT device / IoT device hub may be cumbersome / difficult and error-prone
- The could-based servers or service may get overloaded, may become slow or may be down for some tome; potentially disrupting the functioning of the IoT device or resulting in the loss of some data

Applications of IoT

- Consumer applications
    - Home automation
        - Air conditioners
        - Smart speakers
        - Smart lights
        - Smart switches
        - Security system
            - locks (smart card, biometric — fingerprint, iris, Bluetooth, code)
            - Servillence camera
            - Motion detection camera / lights
        - refridgerators
        - Coffee maker
        - Thermostat
        - Vaccum cleaner robot
- Medical and healthcare
    - Wearables
        - Smart watch
        - Fitness tracker (activity, vitals, sleep)
    - Sensors to monitor vitals for remote health monitoring and telemedicine (pulse rate, SPO2 (Peripheral Oxygen Saturation), ECG (Electrocardiogram), blood pressure, temperature
    - Pacemaker
    - ECG, ventilators, etc.
    -
- Augmented reality / virtual reality
    - Entertainment
    - Information
- Retail
    - Bluetooth beacons
        - Indoor positioning system
        - Indoor position-based advertizing / information service
        - Real-time personalized promotions
    - Inventory optimization
    - Customer tracking and data collection
    - Automated checkout (Amazon Go)
- Energy consumption and environment management
    - Human presence detection
- Factories
    - Real-time monitoring

    - Auto-sensing equipment
    - Machine health monitoring / scheduled maintenance
    - Automated quality control
    - Human health and safety
- Government
    - Smart cities
        - smart grid (online electricity consumption monitoring)
        - water meters (cosumption, leakage)
        - Traffic monitoring
        - Air and water quality monitoring
        - Integrated transport scheduling and management
        - Integrated public transport payment system
        - Online CCTV - e-challan
    - Automated toll collection (FasTAG, GPS-based)
- Autonomous vehicles
- Navigation
- Self-directed vehicles
- Logistics (stock and transport)
    - RFID (Radio Frequency Identification Tag) tags
    - Smart routing of trucks to avoid congestion
- Libraries, hospitals
    - RFID tags
- Vehicles
    - Monitoring of vehicle health
    - Condition-based maintenance
    - Driving monitoring
    - Speed regulator
    - Geo-fencing
    - Vehicle tracking
    - Usage-based insurance rates
- Pharmacy automation
- Gardening
- Agriculture
- Remote Home Control

Smart Assistants

- Google Home (Google Assistant)
- Amazon Echo (Alexa)
- Apple HomePod (Siri)
- Microsoft Windows (Cortana)

Levels of Driving Automation

- **Level 0** The automated system issues warnings and may momentarily intervene but has no sustained vehicle control
- **Level 1 ("hands on")** The driver and the automated system share control of the vehicle. Examples are systems where the driver controls steering and the automated system controls engine power to maintain a set speed (Cruise control) or engine and brake power to maintain and vary speed (Adaptive cruise control or ACC); and Parking Assistance, where steering is automated while speed is under manual control. The driver must be ready to retake full control at any time. Lane Keeping Assistance (LKA) Type II is a further example of Level 1 self-driving. Automatic emergency braking which alerts the driver to a crash and permits full braking capacity is also a Level 1 feature, according to Autopilot Review magazine
- **Level 2 ("hands off")** The automated system takes full control of the vehicle: accelerating, braking, and steering. The driver must monitor the driving and be prepared to intervene immediately at any time if the automated system fails to respond properly. The shorthand "hands off" is not meant to be taken literally — contact

between hand and wheel is often mandatory during SAE 2 driving, to confirm that the driver is ready to intervene. The eyes of the driver might be monitored by cameras to confirm that the driver is keeping their attention to traffic. Literal hands off driving is considered level 2.5, although there are no half levels officially. A common example is adaptive cruise control which also utilizes lane keeping assist technology so that the driver simply monitors the vehicle, such as "Super-Cruise" in the Cadillac CT6 by General Motors or Ford's F-150 BlueCruise

- **Level 3 ("eyes off")** The driver can safely turn their attention away from the driving tasks, e.g. the driver can text or watch a film. The vehicle will handle situations that call for an immediate response, like emergency braking. The driver must still be prepared to intervene within some limited time, specified by the manufacturer, when called upon by the vehicle to do so. You can think of the automated system as a co-driver that will alert you in an orderly fashion when it is your turn to drive. An example would be a Traffic Jam Chauffeur, another example would be a car satisfying the international Automated Lane Keeping System (ALKS) regulations
- **Level 4 ("mind off")** As level 3, but no driver attention is ever required for safety, e.g. the driver may safely go to sleep or leave the driver's seat. However, self-driving is supported only in limited spatial areas (geofenced) or under special circumstances. Outside of these areas or circumstances, the vehicle must be able to safely abort the trip, e.g. slow down and park the car, if the driver does not retake control. An example would be a robotic taxi or a robotic delivery service that covers selected locations in an area, at a specific time and quantities
- **Level 5 ("steering wheel optional")** No human intervention is required at all. An example would be a robotic vehicle that works on all kinds of surfaces, all over the world, all year around, in all weather conditions

SAE (J3016) Automation Levels

<table>
<tr><th>SAE Level</th><th>Name</th><th colspan="2">Narrative definition</th><th>Execution of steering and acceleration/ deceleration</th><th>Monitoring of driving environment</th><th>Fallback performance of dynamic driving task</th><th>System capability (driving modes)</th></tr>
<tr><td colspan="8">Human driver monitors the driving environment</td></tr>
<tr><td>0</td><td>No Automation</td><td colspan="2">The full-time performance by the human driver of all aspects of the dynamic driving task, even when "enhanced by warning or intervention systems"</td><td>Human driver</td><td rowspan="3">Human driver</td><td rowspan="3">Human driver</td><td>n/a</td></tr>
<tr><td>1 (hands on)</td><td>Driver Assistance</td><td>The driving mode-specific execution by a driver assistance system of "either steering or acceleration/deceleration"</td><td rowspan="2">using information about the driving environment and with the expectation that the human driver performs all remaining aspects of the dynamic driving task</td><td>Human driver and system</td><td rowspan="2">Some driving modes</td></tr>
<tr><td>2 (hands off)</td><td>Partial Automation</td><td>The driving mode-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration</td><td>System</td></tr>
<tr><td colspan="8">Automated driving system monitors the driving environment</td></tr>
<tr><td>3 (eyes off)</td><td>Conditional Automation</td><td>The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task</td><td>with the expectation that the human driver will respond appropriately to a request</td><td>System</td><td>System</td><td>Human driver</td><td>Some driving modes</td></tr>
</table>

			to intervene				
4 (mind off)	High Automation		*even if a human driver does not respond appropriately to a request to intervene* the car can pull over safely by guiding system			System	Many driving modes
5 (steering wheel optional)	Full Automation		*under all roadway and environmental conditions* that can be managed by a human driver				All driving modes

IoT Communications Protocols

- **Zigbee** a protocol similar to Bluetooth. It is simpler and less expensive. Hence it is popular among IoT developers
- **Bluetooth Low Energy (Bluetooth LE or BLE)** a version of the Bluetooth protocol that consumes less energy
- **NFC (Near Field Communication)** used for very short range (about 4 cm) communication between cards, mobile devices and POS (Point of Sale) terminals. The so-called *Wi-Fi* credit and debit cards actually use NFC
- **RFID (Radio Frequency IDentification)** used to read information stored in RFID tags. In India, the FASTag tags used for paying highway toll use RFID technology
- **Wi-Fi** used for connecting to the Internet in home / office environments
- **Mobile Communication** GSM (2G), LTE (4G) and 5G can be used for connecting to the Internet from anywhere, particularly useful for outdoor or moving devices
- **Ethernet** wired communication protocol, used to connect the device to a network switch or router
- **PoE (Power over Ethernet)** provides both data communication and power supply using a single Ethernet cable
- **MQTT (Message Queuing Telemetry Transport)** a lightweight protocol for message passing
- **LoRa (Long Range)** / **LoRaWAN** low power, long range proprietary WAN protocol

Concerns

Security

Security is a major concern with IoT because of a variety of factors.

- Any Internet-connected device is susceptible to hacking
- A large number of devices connected to the Internet increase the risk
- As the IoT devices are small with low amount of storage and RAM, the software is highly optimized and uses low-level techniques like pointers and direct memory access
- The hardware and the operating system, if any, have hardly any protection features
- There is a lot of variation in hardware as well ass software tools. Sometimes, the software tools or libraries used may have bugs or vulnerabilities in them

- The communication techniques used are different from mainstream due to limited amount of storage, RAM and processing power. Often, they do not have sufficient security features
- Usually, the software cannot be patched or updated after the device is shipped. Even when updates are possible, the users may not carry them out

Privacy

Often, IoT devices collectively gather a large amount of data. The data are usually sent to could-based servers for processing. There may be misuse of the data. Also, there is a possibility of the data getting stolen during transfer or from the servers. Because of this, the increasing use of IoT raises a lot of security concerns.

The pervasiveness of IoT devices means that data are being collected from practically everywhere. People do not feel comfortable with data about them being collected everywhere.

Data Collection and Monetization

In many cases, a large number of consumer IoT devices collect a lot of data about users. The company collecting the data may use it for advertising or other ways of making money from the data. It may sell or transfer the data to other entities as part of a business deal. This raises privacy concerns.

Arduino

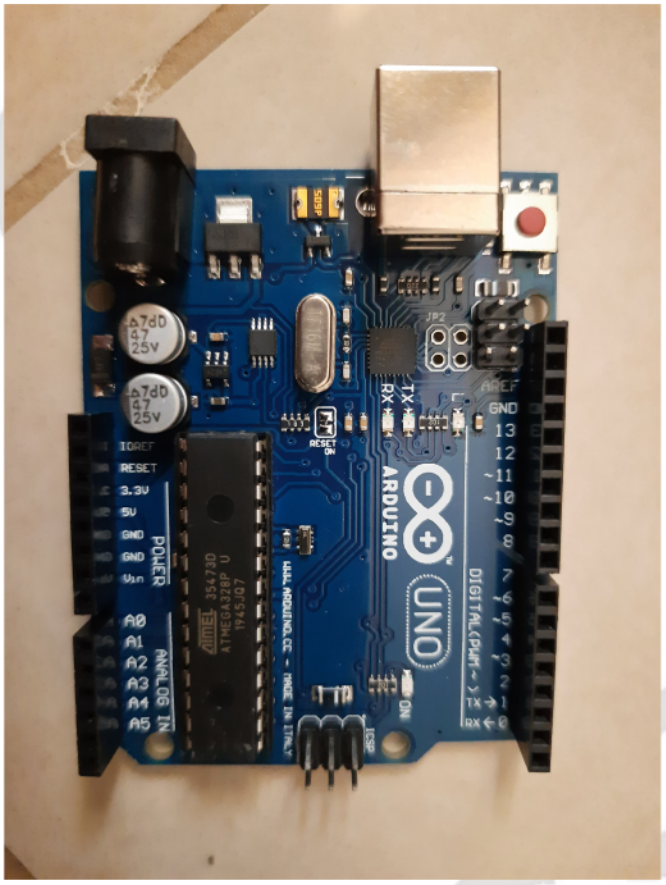
Arduino (/ɑːrˈdwiːnoʊ/) is an open-source hardware and software platform for single-board microcontrollers. There are several Arduino and Arduino compatible boards in the market. Many of them are based on the AVR family of microcontrollers.

Arduino can be powered using a 5V DC power supply (usually using a special USB cable though other methods are also available). It has several pins, including power, ground and GPIO pins. Arduino pins can perform both digital as well as analog input and output. An Arduino communicates with a computer using a serial interface. Usually, the serial interface is emulated using a USB connection and suitable device driver. The Arduino is programmed using this USB-emulated serial interface and it also provides a serial monitor on the same interface for debugging. Being a serial interface, the serial monitor must be configured for the correct Baud Rate.

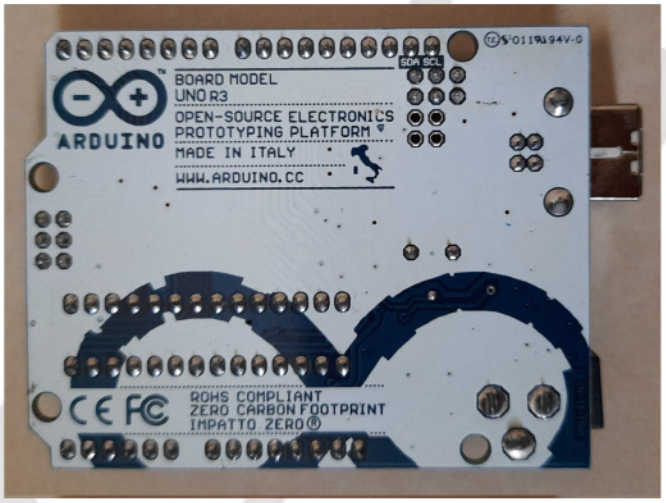
The Arduino Uno

The Arduino Uno is a specific model of the Arduino single-board microcontroller. Its characteristics are as under.

- ATmega328P microcontroller
- 16 MHz CPU frequency
- Required power supply: 5V DC
- 32 KB Flash Memory
- 1KB EEPROM (for the firmware)
- 2KB RAM
- Digital I/O pins: 14
- Analog I/O pins: 6
- V_{in} (power input) pins: 1 (5V DC)
- V_{out} (power output) pins: 1 5V DC, 1 3.3V DC, 2 Ground
- Maximum DC current per I/O pin: 20 mA
- Header type: Female
- No network connectivity
- No real-time clock (RTC), the system time resets at every boot / reset



Arduino Uno R3 Front Side and Pin Layout



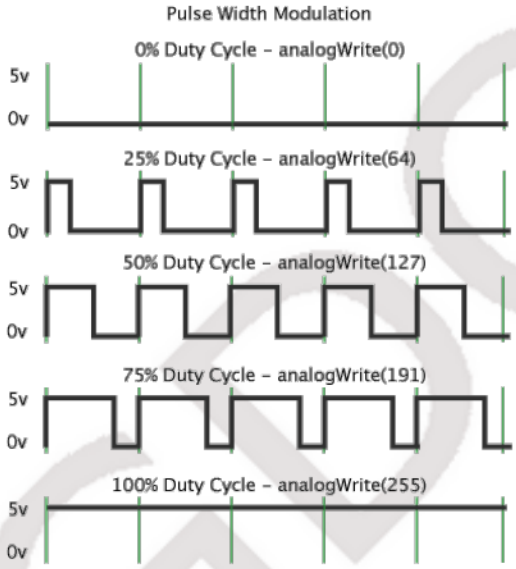
Arduino Uno R3 Back Side

Interfacing with the Arduino

- The digital I/O pins can used for digital input and output. These pins can be in one of the two states only: *HIGH* (5V) and *LOW* (0V / GROUND)
- For analog output, PWM (Pulse Width Modulation) can be used on the digital PWM pins
- For analog input, the analog input pins can be used. These represent the voltage level at the pin (0-5 V) as a number between 0-1023

PWM (Pulse Width Modulation)

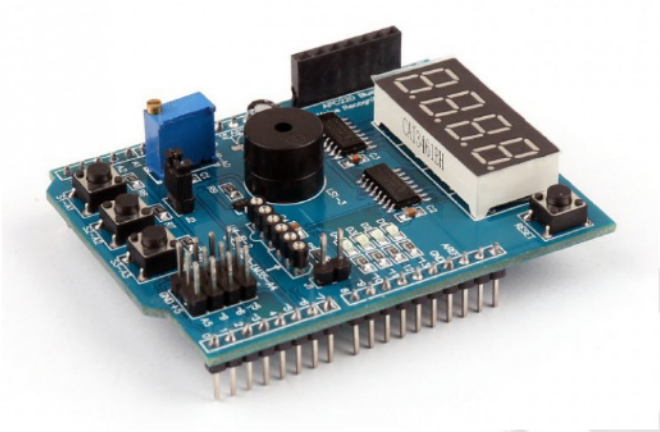
Arduino's digital pins can be in one of the two states only: *HIGH* (5V) and *LOW* (0V / GROUND). To perform analog output, this pin can be switch on and off very fast to give an illusion of a continuously varying output. This is achieved using PWM (Pulse Width Modulation).



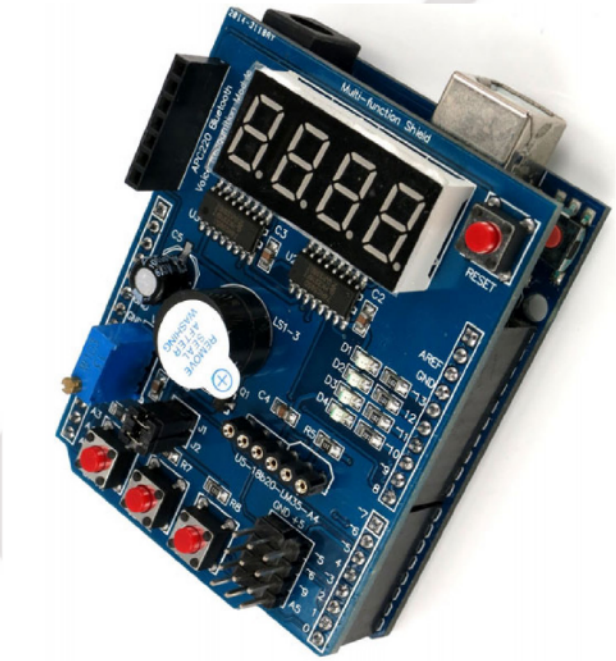
Pulse Width Modulation (PWM)

Arduino Shields

An Arduino shield is an expansion board that can fit into the Arduino pin headers and provide additional functionality like electric motor control, network connectivity, LCD display, SD card slot, real-time clock, etc.



Multifunction Shield for Arduino Uno



Multifunction Shield for Arduino Uno Fitted on Arduino

Arduino Programming

Arduino Studio is a free and open source IDE for programming various Arduino models, third-party Arduino-compatible boards as well as some other embedded systems and IoT devices.

A Program for the Arduino or a compatible device is known as a *sketch*. Sketches are written in C / C++ programming language and use Arduino specific libraries and header files. The filename extension of an Arduino sketch is `.ino`. Only one sketch may be loaded on an Arduino at a time.

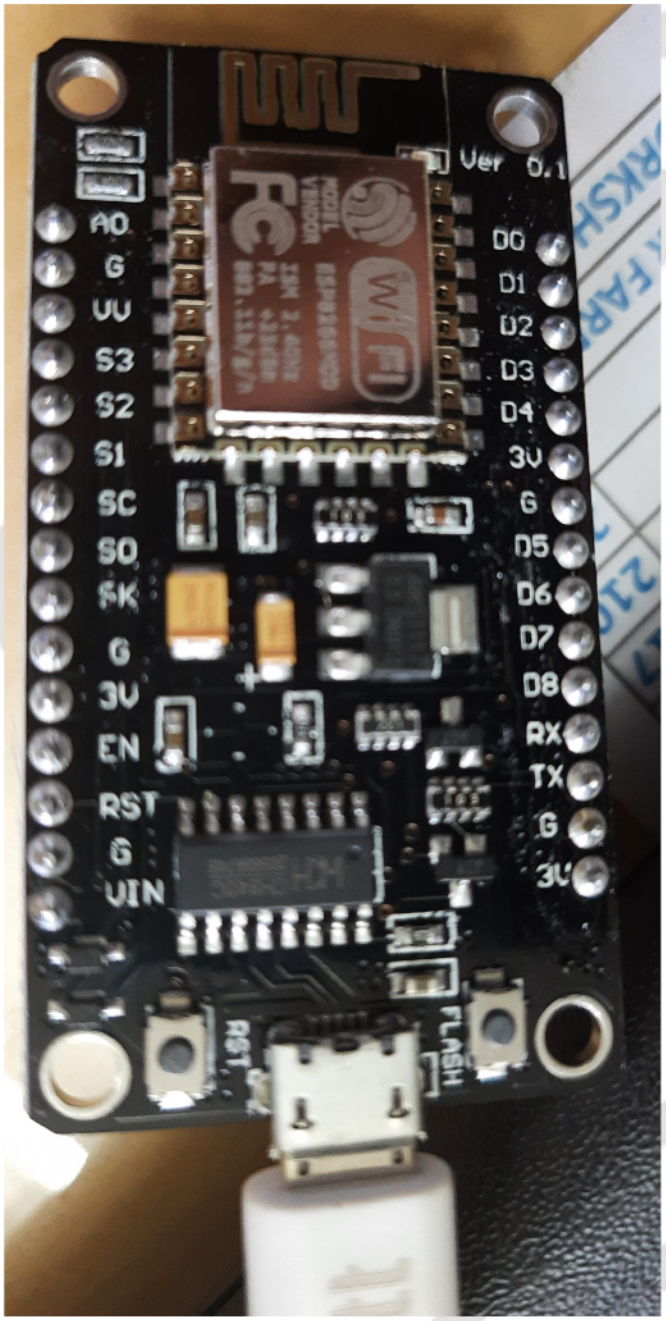
Usually, an Arduino sketch contains a `void setup()` function that runs only once when the Arduino is powered on or reset and a `void loop()` function that keeps running repeatedly (infinite loop) until the power supply to the Arduino is removed or until the Arduino is reset.

```
/*
  Blink

  Turns an LED on for 3 second, then off for 3 second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and
        ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is
        set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your
        Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

//#define BAUDRATE 9600   // Default for Arduino Uno
#define BAUDRATE 115200

// the setup function runs once when you press reset or power the board
void setup() {
  Serial.begin(BAUDRATE);
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the
        voltage level)
  Serial.println("LED On");
  delay(3000);                       // wait for 3 seconds
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the
        voltage LOW
  Serial.println("LED Off");
  delay(3000);                       // wait for 3 seconds
}
```

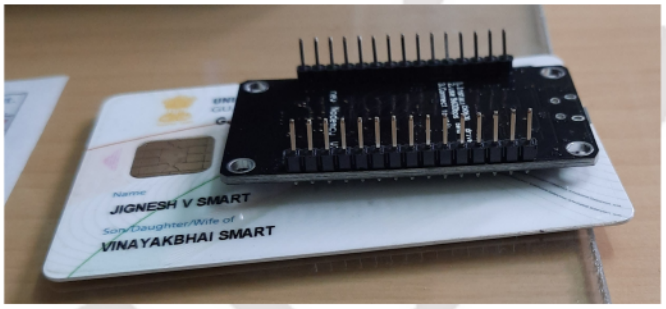
NodeMCU

NodeMCU (MCU stands for Microcontroller Unit) is a low-cost IoT platform that is mostly compatible with the Arduino. While the Arduino does not provide network connectivity by default, the NodeMCU has onboard ESP8266 Wi-Fi with the Wi-Fi antenna etched on the board as a zigzag copper PCB line. Hence, NodeMCU can be used for wireless control and communication with an Arduino-compatible IoT device.

NodeMCU Front Side



NodeMCU Back Side



NodeMCU Side View

The above figure gives a perspective on the size of this SBC.

```
// NodeMCU program for blinking external LED

#define BAUDRATE 115200     // Default baud rate for NodeMCU V3 (?)

#define LED D1

void setup() {
  Serial.begin(BAUDRATE);
  Serial.println();
```

```
  Serial.print("LED number (DEC):");
  Serial.println(LED);
  Serial.print("LED number (HEX):");
  Serial.println(LED, HEX);
  pinMode(LED, OUTPUT);    // LED pin as output.
}

void loop() {
  digitalWrite(LED, HIGH); // turn the LED on.
  Serial.println("LED on");
  delay(3000);            // wait for 3 second.
  digitalWrite(LED, LOW); // turn the LED off.
  Serial.println("LED off");
  delay(3000); // wait for 3 second.
}
```

```
// NodeMCU program for blinking both the builtin and an external LED

#define BAUDRATE 115200     // Default baud rate for NodeMCU V3 (?)

#define BUILTINLED LED_BUILTIN     // value: 2

#define EXTERNALLED D1

void setup() {
  Serial.begin(BAUDRATE);
  Serial.println();
  Serial.print("BUILTINLED number (DEC):");
  Serial.println(BUILTINLED);
  Serial.print("BUILTINLED number (HEX):");
  Serial.println(BUILTINLED, HEX);
  Serial.print("EXTERNALLED number (DEC):");
  Serial.println(EXTERNALLED);
  Serial.print("EXTERNALLED number (HEX):");
  Serial.println(EXTERNALLED, HEX);
  pinMode(BUILTINLED, OUTPUT);    // LED pin as output.
  pinMode(EXTERNALLED, OUTPUT);    // LED pin as output.
}

void loop() {
  digitalWrite(BUILTINLED, HIGH);   // turn the builtin LED off.
                                    // Note that HIGH is the voltage level
        but actually
                                    // the LED is turned off;
                                    // this is because it is acive low on
        the ESP8266.

  digitalWrite(EXTERNALLED, HIGH);  // turn the external LED on.

  Serial.println("BUILTIN LED off, EXTERNALLED on");

  delay(5000);                      // wait for 5 second.

  digitalWrite(BUILTINLED, LOW);    // turn the builtin LED on.
                                    // Note that LOW is the voltage level
        but actually
                                    // the LED is turned on;
                                    // this is because it is acive low on
        the ESP8266.

  digitalWrite(EXTERNALLED, LOW);   // turn the external LED off.

  Serial.println("BUILTIN LED on, EXTERNALLED off");

  delay(5000); // wait for 5 second.
}
```

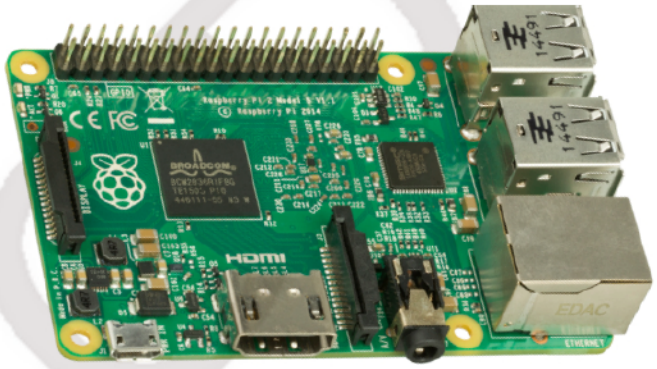
```
// Fading in and out LED brightness using PWM

#define BAUDRATE 115200     // Default baud rate for NodeMCU V3 (?)

const int PWMPin = 3;

void setup()
{
   Serial.begin(115200);
   Serial.println("PWM Fade in fade out...");
}
```

```
void loop()
{
static int pwm;
static float x = 0;
static int step = 2;
static bool increasing = true;

   if (increasing) {
    x += step;
    if (x > 255) {
        increasing = false;
        x = 255;
      }
   } else {
    x -= step;
    if (x < 0) {
      increasing = true;
      x = 0;
    }
   }

   analogWrite(PWMPin, x);

   delay(50);
}
```

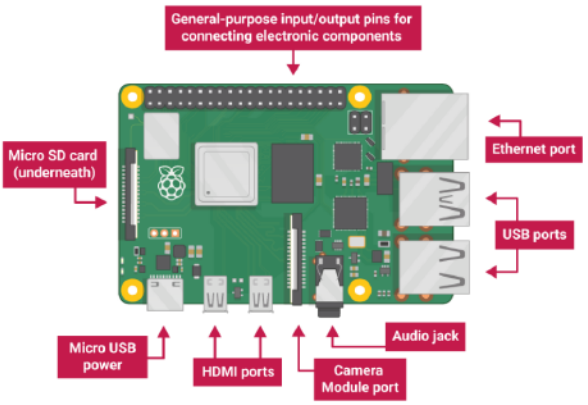
Raspberry Pi

Introduction

Raspberry Pi (RPi or Pi) is a comparatively larger single-board computer (SBC) developed by the Raspberry Pi Foundation, UK. It uses a Broadcom SoC. It was aimed at teaching computer programming to school children. However, it has become quite popular among hobbyists as well as commercial product makers. The target price for the base model was US $35. The Foundation has released several models of the Raspberry Pi with different capabilities. The latest models are in the Raspberry Pi 4 series. A number of operating systems can be installed on the Pi and different programming languages can be used for programming it. It supports both CLI and GUI.



Raspberry Pi 2 Model B



Raspberry Pi connectors

Raspberry Pi 2 Model B

- SoC : Broadcom BCM2836 Quad code ARM Cortex-A7 32-bit SoC @ 900 MHz
- RAM : 1GB
- Storage : Micro-SD card slot
- Power supply : 5V DC (Micro USB)
- Form factor (size) : Standard
- GPIO pins in the header: 40
- USB ports : 4 × USB 2.0 ports
- Network connectivity : 1 10/100 Ethernet port, no Wi-Fi or Bluetooth
- Display output : HDMI port
- Audio output: 3.5 mm jack
- Camera (webcam) support : specialized camera

Raspberry Pi 4 Model B

- SoC : Broadcom BCM2711 Quad core ARM Cortex-A72 64-bit SoC @ 1.5GHz
- RAM : 1GB / 2GB / 4GB / 8GB
- Storage : Micro-SD card slot
- Power supply : 5V DC (USB Type C)
- Form factor (size) : Standard
- GPIO pins in the header: 40
- USB ports : 2 × USB 2.0, 2 × USB 3.0
- Network connectivity : 1 10/100/1000 Gigabit Ethernet port, onboard Wi-Fi, onboard Bluetooth
- Display output : 2 × Micro-HDMI ports
- Audio output: 3.5 mm jack (4-pole stereo)
- Camera (webcam) support : specialized camera

Operating Systems and Software

The operating system for the Raspberry Pi is installed by downloading the disk image on a computer and then writing it to an SD card. The SD card is then inserted in the Pi and is used to boot it.

A large number of operating systems are available for the Raspberry Pi. Some support only CLI (Command Line Interface), while others support full graphical interface.

- Raspberry Pi OS (earlier called Raspbian) -this Debian Linux based OS is the official OS for the Raspberry Pi
- Ubuntu, Ubuntu Mate
- Windows 10 IoT Core (free, text mode only)
- LibreELEC (Used to run the Kodi media player)

Once the operating system is installed, additional software can be installed using the operating system's software installer. Commonly, Python is used to write programs for the Raspberry Pi.

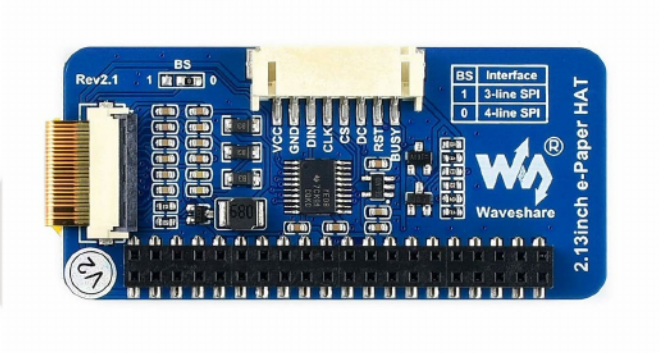
Interfacing with the Raspberry Pi

- The GPIO pins on the Pi can used for digital input and output. These pins can be in one of the two states only: *HIGH* (3.3V) and *LOW* (0V / GROUND)
- For analog output, PWM (Pulse Width Modulation) can be used on the digital PWM pins
- There is no built-in support for analog input

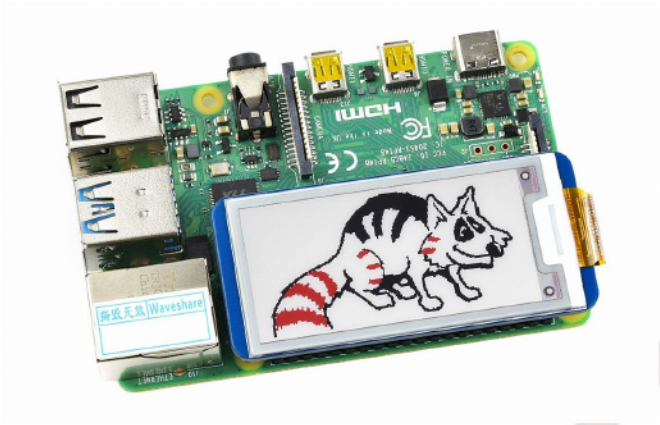
Raspberry Pi Hats

A Raspberry Pi Hat is an expansion board that can fit into the Rspberry Pi pin headers and provide additional functionality.

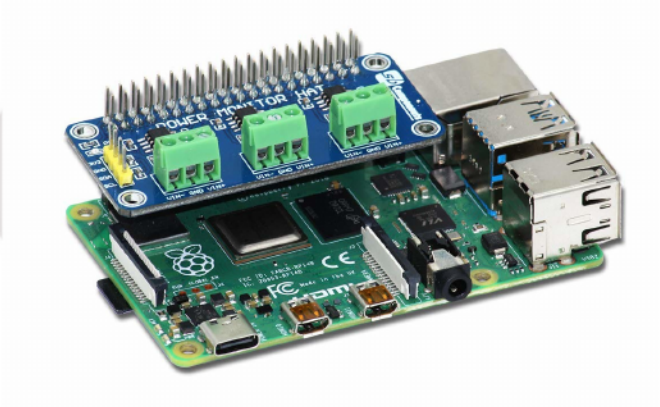
Raspberry Pi e-paper Display Hat-1



Raspberry Pi e-paper Display Hat-2



Raspberry Pi e-paper Display Hat-3



Raspberry Pi 3-channel Current/Voltage/Power Monitoring Hat

Raspberry Pi Programs

```
# Blink an LED

from gpiozero import LED
from time import sleep

led = LED(14)

while True:
    print('Turning LED On...')
    led.on()
    sleep(2)
    print('Turning LED Off...')
    led.off()
    sleep(2)
```

```
# Controlling multiple LEDs, or
# an RGB LED and a single-color LED

import sys
from gpiozero import LED
from time import sleep

print('sys.path:')
print(*sys.path)
print()

led14 = LED(14)
led15 = LED(15)
led18 = LED(18)
led17 = LED(17)
delay=5

while True:
    print('Turning LED-14 On...')
    led14.on()
    sleep(delay)
    print('Turning LED-14 Off...')
    led14.off()
    print('Turning LED-15 On...')
    led15.on()
    sleep(delay)
    print('Turning LED-15 Off...')
    led15.off()
    print('Turning LED-18 On...')
    led18.on()
    sleep(delay)
    print('Turning LED-18 Off...')
    led18.off()
    print('Turning LED-17 On...')
    led17.on()
    sleep(delay)
    print('Turning LED-17 Off...')
    led17.off()
    sleep(delay)
```

```python
# Controlling the brightness of an LED using PWM

from gpiozero import PWMLED
from time import sleep

led = PWMLED(17)

i=0
while True:
    value = i / 10
    print('LED value:', value)
    led.value = value
    led.on()
    sleep(2)
    print('Turning LED Off...')
    led.off()
    sleep(2)
    i = i + 1
    if i == 10:
        i = 0
```

Electronics

Power supply

- AC (Alternating Current) ~
  - **Wall socket supply** 230V AC
- DC (Direct Current) ⎓
  - **Cells**
  - **Batteries**

AC to DC Conversion

```
- Adapter (input voltage, input current, output voltage)
+ Raspberry Pi 2: Input: 230V ~, 2A Output: 5V ⎓
    + Charger
```

Voltage, Current and Resistance

- Voltage
  - Electromotive force (EMF)
  - Unit: Volt (V)
- Current
  - Stream of negatively charged particles (electrons). However, the convention is to draw it as a flow of positively charged particles in circuits
  - Unit: Ampere (A), Milliampere (mA)
- Resistance
  - Resistance against flow of electrons
  - Unit Ohm (Ω)

Ohm's Law

Specifies relationship between the voltage (V), current (I) and resistance (R) in a DC circuit.

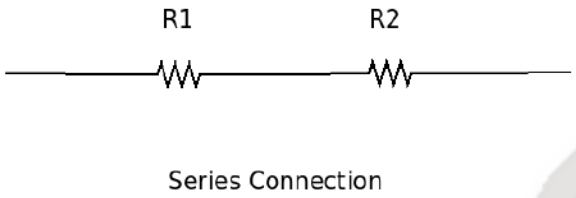
$$V = I \times R$$

or,

$$I = \frac{V}{R}$$

Series v/s Parallel Connections

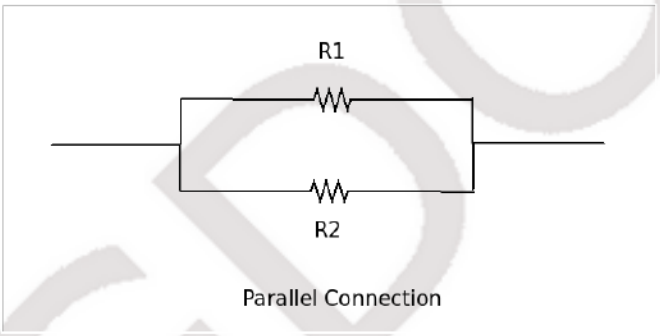
Series connections



Series Connection

```
R = R1 + R2
```

Parallel connections



Parallel Connection

$$\frac{1}{R} = \frac{1}{R1} + \frac{1}{R2}$$

$$R = \frac{R1 \; R2}{R1 + R2}$$

Resistor

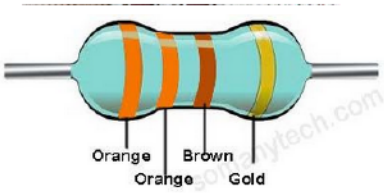


Fig. showing color code of 330 ohm resistor

[Orange, orange, brown, gold]

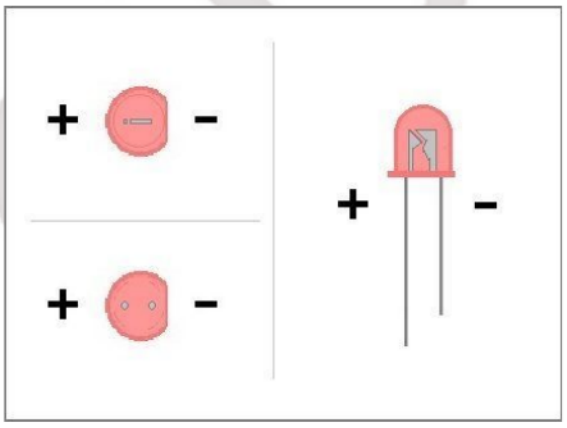
Band	Color	Value
1st Orange		3
2nd Orange		3
3rd Brown		10
4th gold		+-5%
	1st digit 3 2nd digit 3 3rd multiplier: 10	330 Ohm tolerance: +-5%

Resistor Color Code

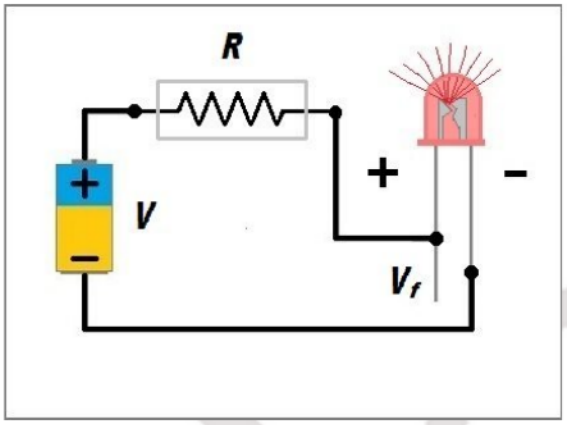
LED (Light Emitting Diode)

An LED is a diode. It allows current to pass only in one direction. When current passes through the LED, it emits lights or glows. Hence it is known as Light Emitting Diode. An LED has two leads - *cathode* and *anode*. Usually, the anode lead is slightly longer than the cathode lead. Conventional (positive) current only flows from the anode to the cathode. Hence the anode must be connected to the positive voltage, while the cathode must be connected to the negative voltage or ground.

Since the LED has a very low resolution in the forward bias configuration, it is necessary to connect a resistor (~ 330 Ω) in series with the LED to prevent it from drawing too much current.



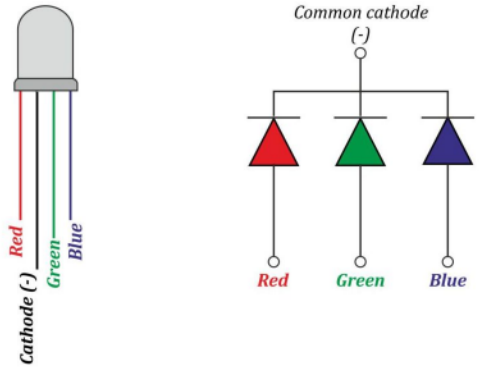
LED Leads Polarity



LED Connection

Multi-color (RGB) LED

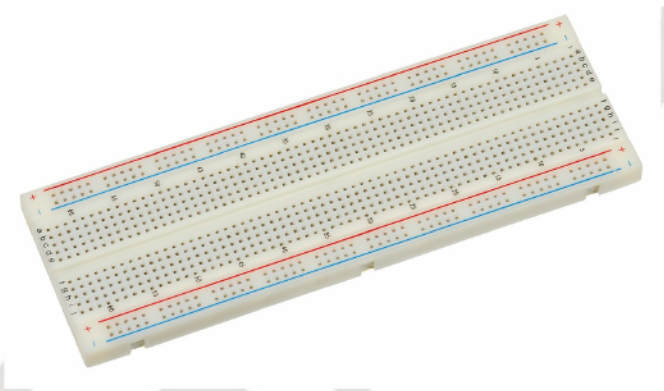
A multi-color LED has multiple LEDs of different colors inside a single package. The LEDs of different colors can be turned on and off individually. For example, an RGB LED has three LEDs inside it - Red, Green and Blue. All the LEDs have one lead in common. In a common cathode LED, the cathode is common and there is a separate anode to control each individual LED.



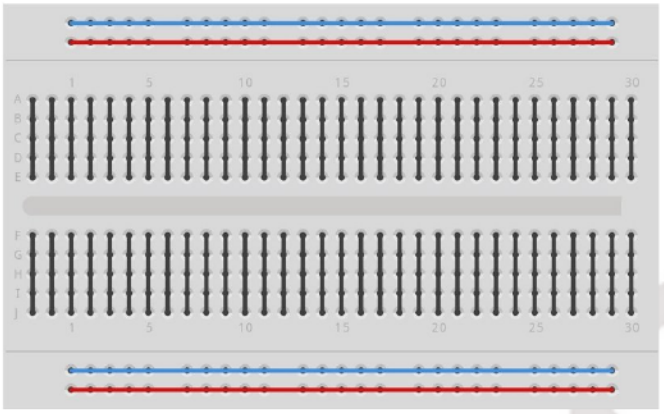
RGB LED

Breadboards and PCBs

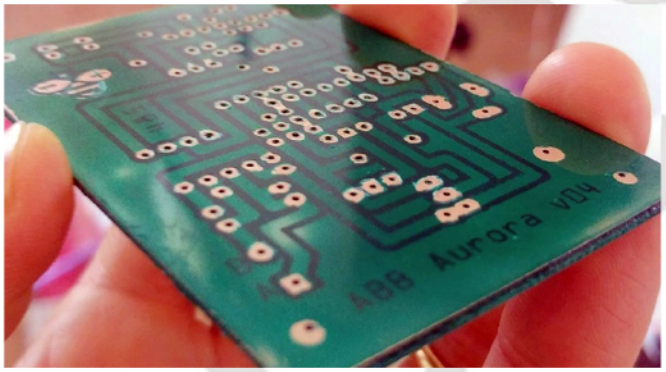
A breadboard, or protoboard, is a construction base for creating temporary prototypes of electronic circuits for testing and learning. Once the circuit is designed and tested successfully, its final version may be created in PCB (Printed Circuit Board).



Breadboard



Internal Connections in a Breadboard



PCB

Transistors

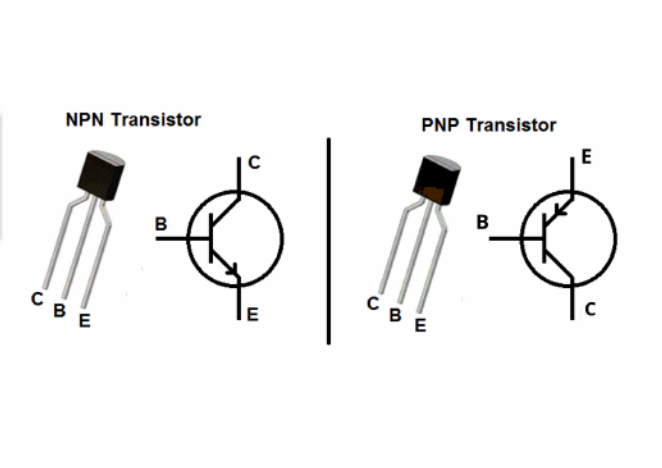
A transistor is a semiconductor device used amplify or switch electrical signals. With a transistor, a small amount of DC current can be used to control a comparatively larger DC current.

- **Transistor as an amplifier** A transistor can be used to turn a weak signal into a stronger signal
- **Transistor as a switch** Using a transistor, a small amount of current can be used to turn a comparatively larger flow of current on or off.

There are many types of transistors. The two common categories are `NPN` and `PNP`. They have different uses.

- Transistor Terminals
  - Collector (C)
  - Base (B)
  - Emitter (E)

A small amount of current between the base and the emitter can be used to control a large amount of current between the collector and emitter.

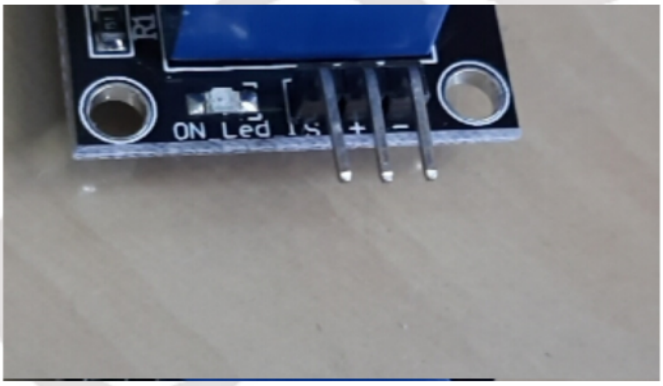


NPN and PNP Transistors

Relays

A relay is an electro-mechanical switch that can be used to control very high amount of AC / DC current using a small amount of DC current.

A Relay

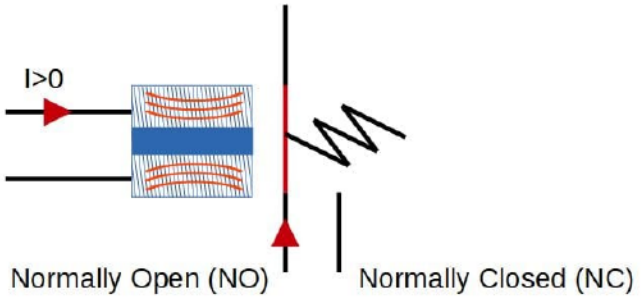


Low-power Side Terminals of a Relay

The `+` terminal must be supplied 5V DC while the `-` terminal must be supplied 0V DC (Ground). The `S` (signal) terminal is used to turn the relay on or off.

High-power Side Terminals of a Relay

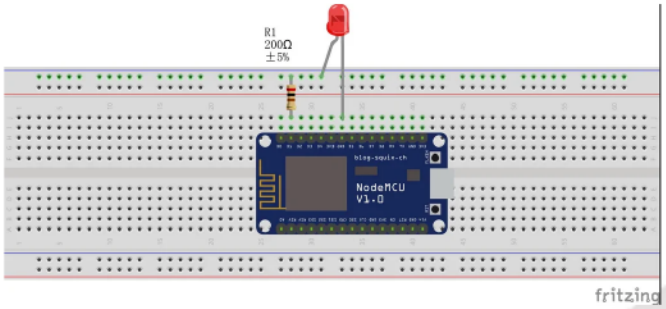
By default (in the absence of signal in the low-power circuit), the central terminal is connected with the `NC` (Normally Closed) terminal but not with the `NO` (Normally Open) terminal. When there is signal in the low-power circuit, this is reversed. The central terminal is now connected to the `NO` terminal, but not the `NC` terminal.



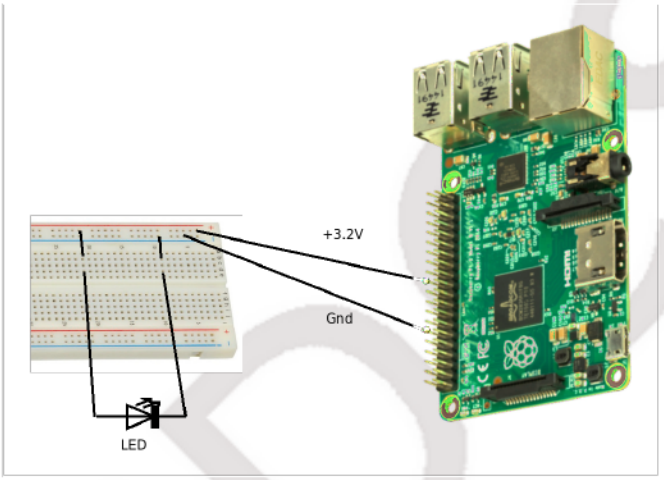
Relay Operation

The relay is an electromagnetic switch that operates using a permanent magnet, an electromagnet (coil) and a spring. When it changes state, there is a characteristic *click* sound. (Other types of relays are also available)

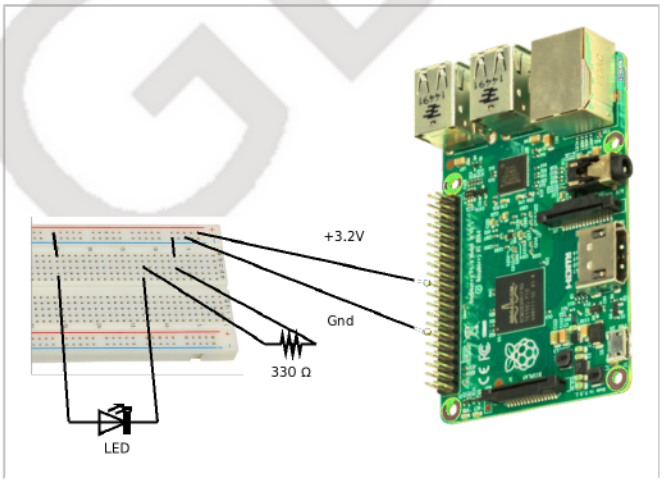
Projects



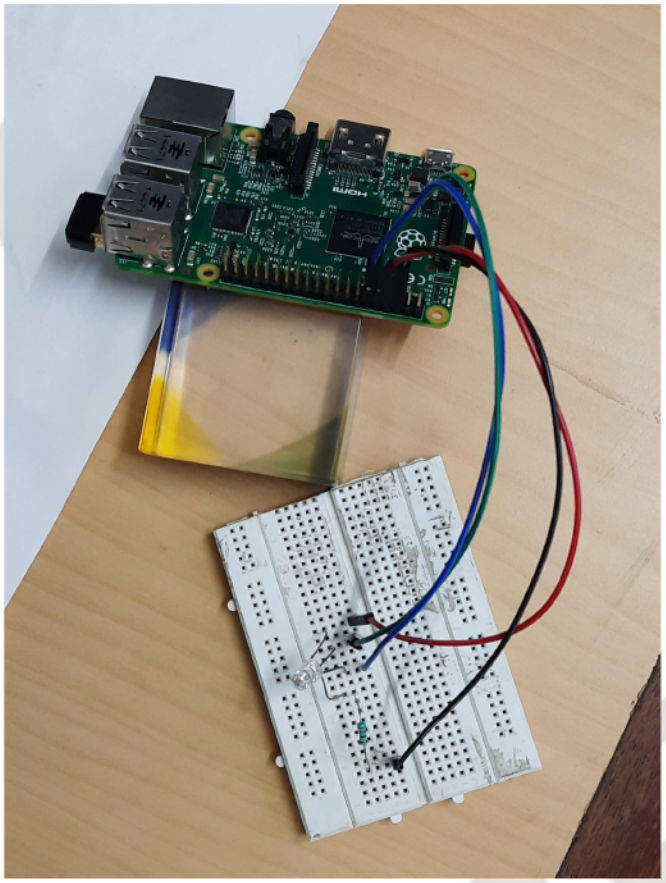
Arduino External LED Circuit



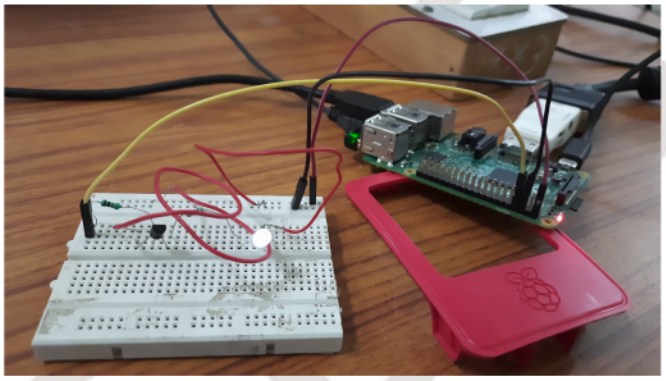
Raspberry Pi LED Circuit (Without Resistor)



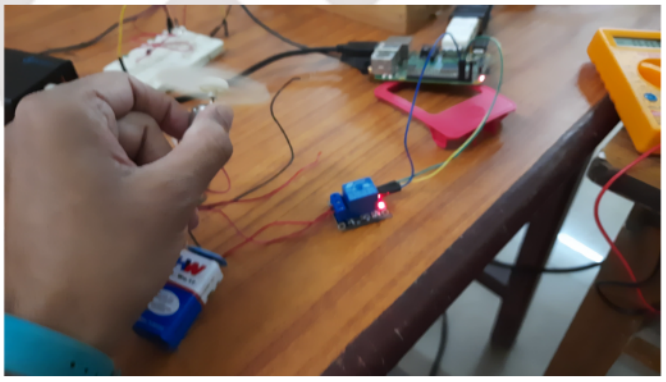
Raspberry Pi LED Circuit (With Resistor)



Raspberry Pi RGB LED Circuit



Controlling an LED using a transistor



Controlling an electric motor using a relay