

Higher Future Institute of
Technological Studies
of the Specialized
INFORMATION SYSTEMS DEPARTMENT



SmartX
X AUTOMATION SYSTEM

By:

Mohamed Mostafa Elghabaty
Mahmoud Salah El-Din Lasheen
Khaled Elsayed Hassan
Mohamed Yousef Ramadan
Mostafa Fawzy Anwar
Mohamed Sedeek Raslan

Supervised By:

Eng. Dr. \ MOHAMMED AMEER

2017



By:

Mohamed Mostafa Elghabaty

Mahmoud Salah El-Din Lasheen

Khaled Elsayed Hassan

Mohamed Yousef Ramadan

Mostafa Fawzy Anwar

Mohamed Sedeek Raslan

Supervised By:

Eng. Dr. \ MOHAMMED AMEER

Acknowledgments

First, we have to express our gratitude and appreciation to all of our doctors who has suffered with us along the last four years .and we cannot forget all our instructors too.

In our project that would not become known without the support of some people, these people start with

A special one who was our mentor and our director who gave us the opportunity to learn and benefit us from his knowledge this person is:

Eng. Dr. / Mohamed Ameer

Who honestly has leaved a good impression on us and treated us as one of his sons and always encouraged us to do more.

We can't forget to thank our friends in our department for their effort and helping us in our project.

At last we would thank ourselves to adapt with working in team work and bearing each other in order to produce this project in fine way.

Abstract

Home Automation is a way to have things around your home happen automatically. The first thing that comes to mind when folks think of home automation are robots, flashing lights, complicated electronics and a general feeling that their home is less of a warm home and more of a cold science experiment. However, in most homes today, you can easily find some simple forms of automation such as: Garage door openers, Remote Controls, Irrigation / sprinkler control systems, Motion activated lights, Security systems, Programmable thermostats, Programmable light timers If you want to keep going, you can throw in dishwasher, clothes washers and dryers, ovens, microwaves, cars, lights and switches.... The list goes on and on. You may not think of a dishwasher or light switch as home automation, but compared to washing dishes by hand and striking a match to light a candle every time you enter a room, it's defiantly automation. However, each of these things was designed to help us do some complicated, strenuous, unpleasant, or repetitious action automatically. The term 'Home Automation' today applies to the next level of automating home electronics.

Automate your home

To make life easier. We invented TV remote controls so we didn't have to get out of the chair to change the channel. Some people now own complex media systems that require the owner to press 10 different buttons on 5remotes just to watch Oprah.

You can:

Press one button on a remote control and have it dim the lights, set the volume level, and start playing a movie and music which you desire. Not have the sprinklers turn on if it just rained. Get emails sent to you at work or on your cell phone if a motion detector or security system is tripped while out of the house. Get emails sent to you with the caller ID information of a call received at your house when out. Automatically turn the lights on the house when the garage door goes up and it's after sundown. Automatically turn the front porch lights on 1/2 hour before sundown every day (and automatically adjust for daylight savings). Automatically close the garage doors every night. Automatically turn on holiday lights at specific times (all at once). The full list is limited to imagination and a family's lifestyle.

Goal

Provide convenience for the owner. Additionally, value can be realized by using automation to save energy and to provide security and peace of mind. Finally, we all appreciate the fun and excitement of experiencing the latest technology in our home or business.

While there are many home automation solutions available today, they vary widely in terms of affordability, reliability and most importantly with their ability to withstand the all-important test of time. It is true that everything you install today will at some point in the future be considered obsolete.

List of figures

Figure (1.1): SmartX X1 device design

Figure (1.2): Android app version 1.5 design

Figure (2.1): Characteristics of an embedded system

Figure (2.2): Basic structure of an embedded system

Figure (2.3): IOT Map

Figure (3.1): Raspberry pi 3

Figure (3.2): Raspberry pi 3 hardware

Figure (3.3): Arduino mega

Figure (3.4): Screenshot of the Arduino IDE showing the Blink simple
beginner program

Figure (4.1): Diode

Figure (4.2): The current-voltage relationship of an ideal diode

Figure (4.3): Diode cathode identification

Figure (4.4): Diode types

Figure (4.5): Schottky diodes isolated circuit

Figure (4.6): Breadboard

Figure (5.1): PIR sensor

Figure (5.2): PIR Detection

Figure (5.3): Photoresistor

Figure (5.4): Ultrasonic transducer

Figure (5.5): Sound field of a non-focusing 4 mhz ultrasonic transducer
with a near field length of $N = 67$ mm in water

Figure (4.6): Sound pressure field of the same ultrasonic transducer
(4 mhz, $N = 67$ mm) with the transducer surface having a spherical
curvature with the curvature radius $R = 30$ mm

Figure (5.7): Smoke detector sensor

Figure (6.1): Relay module

Figure (6.2): Xbee module

Figure (7.1): i2c design

Figure (8.1): Gas sensor pinout

Figure (8.2): Ultra-sonic sensor pinout

Figure (8.3): Temp sensor pinout

Table of Contents

| | |
|--|----|
| List of figures | IV |
| Table of Contents..... | V |
| Chapter (1): Introduction | 1 |
| 0.1 SmartX: | 1 |
| 0.1.1 How SmartX Works?..... | 1 |
| 0.1.2 Device:..... | 1 |
| 0.1.3 Mobile App:..... | 2 |
| 0.1.4 How to Contribute:..... | 2 |
| 1.1 Home Automation: | 3 |
| 1.1.1 Lighting..... | 3 |
| 1.1.2 HVAC..... | 3 |
| 1.1.3 Maintenance..... | 4 |
| 1.1.4 Safety..... | 4 |
| 1.1.5 Costs..... | 4 |
| Chapter (2): Survey | 5 |
| 2.1 Embedded Systems | 5 |
| 2.1.1 Characteristics of an Embedded System..... | 5 |
| 2.1.2 Advantages | 7 |
| 2.1.3 Disadvantages | 7 |
| 2.1.4 Basic Structure of an Embedded System..... | 7 |
| 2.2 The Internet of Things | 8 |
| Internet of Things | 9 |
| Impact | 9 |
| Chapter (3): Microcontrollers | 10 |
| 3.1 Raspberry Pi | 11 |
| 3.1.1 Overview | 11 |
| 3.1.2 Hardware | 13 |
| 3.1.3 Processor | 13 |
| 3.1.4 Performance..... | 14 |

| | |
|---|-----------|
| 3.1.4 Overclocking..... | 14 |
| 3.1.5 RAM | 16 |
| 3.1.6 Networking | 16 |
| 3.1.7 Peripherals..... | 16 |
| 3.1.8 Video..... | 17 |
| 3.1.9 Real-time clock | 18 |
| 3.2 Arduino..... | 19 |
| 3.2.1 Overview | 19 |
| 3.2.2 Hardware | 21 |
| 3.2.3 Shields..... | 22 |
| 3.2.4 Software..... | 23 |
| 3.2.5 Arduino IDE..... | 24 |
| Chapter (4): Electronics-components | 25 |
| 4.1 Diode:..... | 25 |
| 4.1.1 Introduction..... | 25 |
| 4.1.2 Ideal Diodes..... | 25 |
| 4.1.3 Types of Diodes..... | 26 |
| 4.2 Light-emitting diode (LED):..... | 28 |
| 4.3 Resistors:..... | 30 |
| 4.4 Ohm's Law:..... | 31 |
| 4.5 Capacitor:..... | 32 |
| 4.5.1 Capacitor symbols..... | 32 |
| 4.5.2 Capacitance | 33 |
| 4.5.3 Capacitance of plates capacitor | 33 |
| 4.6 Transistor:..... | 34 |
| 4.7 Switch: | 35 |
| 4.8 Relay:..... | 36 |
| 4.8.1 Relay Construction..... | 36 |
| 4.9 Breadboard:..... | 37 |
| Chapter (5): Sensors | 39 |
| 5.1 Pyroelectric ("Passive") InfraRed sensors: | 39 |
| 5.1.1 Operation..... | 39 |

| | |
|---|----|
| 5.2 Photoresistor | 41 |
| 5.2.1 Design considerations | 42 |
| 5.3 Ultrasonic transducer | 43 |
| 5.3.1 Capabilities and limitations..... | 44 |
| 5.3.2 Transducers..... | 45 |
| 5.4 Smoke Detector | 47 |
| Chapter (6): Modules | 48 |
| 6.1 Relay Module: | 48 |
| 6.2 Xbee: | 49 |
| Chapter (7): Protocols | 50 |
| 7.1 I²C | 50 |
| 7.1.1 Design..... | 50 |
| 7.1.2 Message protocols..... | 51 |
| 7.1.3 Circuit interconnections..... | 52 |
| Chapter (8): Implementation | 53 |
| SERIAL CONNECTION: | 53 |
| Android “Java”:..... | 53 |
| Arduino C: | 54 |
| Python:..... | 54 |
| CONTROL MODES “Manual - Auto”: | 55 |
| Android “Java”:..... | 55 |
| Arduino C: | 55 |
| Python:..... | 55 |
| SEND & RECEIVE BYTES: | 56 |
| Android “Java”:..... | 56 |
| Arduino C: | 56 |
| Python:..... | 56 |
| ROOMS STATUS: | 57 |
| Android “Java”:..... | 57 |
| Arduino C: | 58 |

| | |
|----------------------------------|----|
| GAS SENSOR: | 59 |
| Android "Java": | 59 |
| Arduino C: | 60 |
| Python: | 60 |
| MOTION SENSOR: | 61 |
| Android "Java": | 61 |
| Arduino C: | 62 |
| TEMPERATURE SENSOR: | 63 |
| Android "Java": | 63 |
| Arduino C: | 64 |
| Python: | 64 |

Chapter (1): Introduction

Automation can make things easier, safer, and often more cost efficient. What was once the stuff of world's fairs and science fiction being today a reality, so learn about the benefits and capabilities of applying today's technology to your home.

0.1 SmartX:

Put any word after smart ... it will be (smart home, smart company, smart factory, etc.) Here is how we get the idea to make a one device that can control deferent places without any extra components.

0.1.1 How SmartX Works?

SmartX is a open source platform based on Arduino and Raspberry Pi 3, with a android application that allows you to control or monitor your place remotely.

0.1.2 Device:

Because we want to make a one device, we made a simple case that protect the inside components and implements all the microcontrollers devices with a monitoring LCD to check the status.



Figure (1.1): SmartX X1 device design

0.1.3 Mobile App:

To make things easier we made a mobile application that allows you to control your place or apartment remotely.

Also you can view your place history with date and time (In case of travel) "future feature".



Figure (1.2): Android app version 1.5 design

0.1.4 How to Contribute:

Simply go to this link <https://goo.gl/psTjSV> and you will find all the open source materials.

1.1 Home Automation:

Home automation is a general term that covers a variety of technological capabilities you can install in your home. Home automation can include controlling aspects of your home remotely through a computer or phone, programming electronic devices to respond automatically to certain conditions or scenarios, or centralizing the control of a variety of items in your home into a single control center.

1.1.1 Lighting

Among the popular applications of home automation is lighting. Not only can you tweak the lighting controls and personalize their output to your needs, but you can also save money by using the lighting more efficiently.

Control all the lights in and around your house from one central location. There's no need to get out of bed or go downstairs if you forgot to turn off a light – just do it remotely!

Arrange for specific lights to go on and off at preset times. Have your lights go on when it's time to wake up, turn on outdoor lighting when it gets dark, or have your lights cycle on and off when you're on vacation.

Control your home's lighting from a distance. If you're returning home late at night and want your house illuminated before you arrive, just adjust the lights over the internet and save electricity.

1.1.2 HVAC

In cold cities, it's common for people to leave their heating on a low temperature during the day so the house doesn't get too cold. While this may be convenient, it's quite costly. Instead, you can set your heating to go off when you leave your home and have it come on and warm up the house 15 minutes before you arrive. If your plans change, you can reset the system from your office or while you're on the way home.

1.1.3 Maintenance

Not only does home automation make your home more efficient and easier to operate, it can also help you keep it maintained. Appliances can be connected to devices that monitor their activity; so a self-monitoring furnace, for example, can tell you when it needs cleaning, and an air conditioner can report that it needs a new filter.

1.1.4 Safety

Home automation can also be used to effectively protect you and your loved ones. Beyond traditional smoke detectors and security systems, you can interconnect the technology in your home to awaken you with lights and sounds if a smoke alarm goes off, no matter what room you're in. In addition, you can have the same system shut down all other audio and video devices in the home to eliminate distractions, and have it notify the fire department at the same time – all without you doing a thing.

1.1.5 Costs

Because home automation has virtually limitless possibilities, it's difficult to put a price tag on it. You can have an electrician install an all-in-one system for anywhere between \$10,000 and \$20,000, which includes a flat screen TV, computer, digital thermostat, dimmer switches, wireless video cameras and more. Alternatively, if you only want to automate certain features or parts of your home, you can expect to pay an electrician roughly between \$3 and \$5 per square foot, depending on the features and accessories you select.

Chapter (2): Survey

2.1 Embedded Systems

As its name suggests, Embedded means something that is attached to another thing. An embedded system [\[1\]](#) can be thought of as a computer hardware system having software embedded in it. An embedded system can be an independent system or it can be a part of a large system. An embedded system is a microcontroller or microprocessor based system which is designed to perform a specific task. For example, a fire alarm is an embedded system; it will sense only smoke.

An embedded system has three components –

- It has hardware.
- It has application software.
- It has Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a plan to control the latencies. RTOS defines the way the system works. It sets the rules during the execution of application program. A small scale embedded system may not have RTOS.

So we can define an embedded system as a Microcontroller based, software driven, reliable and real-time control system.

2.1.1 Characteristics of an Embedded System

- Single-functioned – an embedded system usually performs a specialized operation and does the same repeatedly. For example: A pager always functions as a pager.
- Tightly constrained – All computing systems have constraints on design metrics, but those on an embedded system can be especially tight. Design metrics is a measure of an implementation's features such as its cost, size, power, and performance. It must be of a size to fit on a single chip, must perform fast enough to process data in real time and consume minimum power to extend battery life.

Chapter(2):
Survey

- Reactive and Real time – Many embedded systems must continually react to changes in the system's environment and must compute certain results in real time without any delay. Consider an example of a car cruise controller; it continually monitors and reacts to speed and brake sensors. It must compute acceleration or de-accelerations repeatedly within a limited time; a delayed computation can result in failure to control of the car.
- Microprocessors based – It must be microprocessor or microcontroller based.
- Memory – It must have a memory, as its software usually embeds in ROM. It does not need any secondary memories in the computer.
- Connected – It must have connected peripherals to connect input and output devices.
- HW-SW systems – Software is used for more features and flexibility. Hardware is used for performance and security.

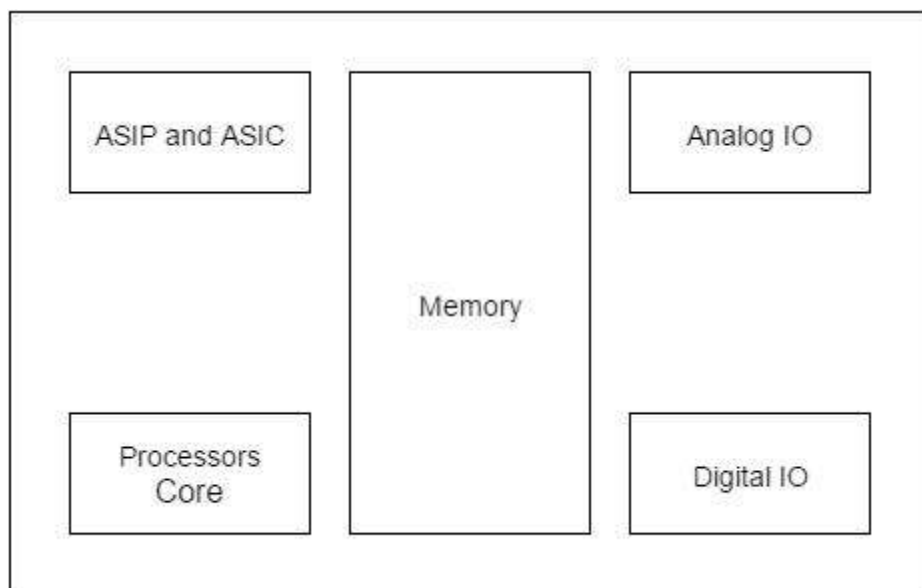


Figure (2.1): Characteristics of an embedded system

2.1.2 Advantages

- Easily Customizable
- Low power consumption
- Low cost
- Enhanced performance

2.1.3 Disadvantages

- High development effort
- Larger time to market

2.1.4 Basic Structure of an Embedded System

The following illustration shows the basic structure of an embedded system

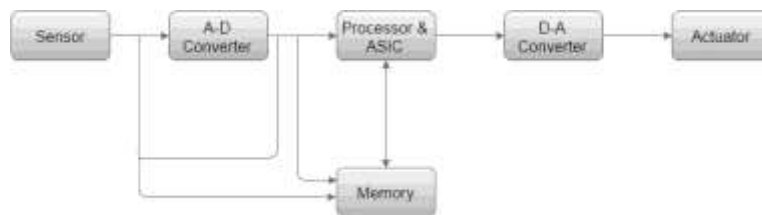


Figure (2.2): Basic structure of an embedded system

–Sensor – It measures the physical quantity and converts it to an electrical signal which can be read by an observer or by any electronic instrument like an A2D converter. A sensor stores the measured quantity to the memory.

- A-D Converter – an analog-to-digital converter converts the analog signal sent by the sensor into a digital signal.
- Processor & ASICs – Processors process the data to measure the output and store it to the memory.
- D-A Converter – A digital-to-analog converter converts the digital data fed by the processor to analog data
- Actuator – An actuator compares the output given by the D-A Converter to the actual (expected) output stored in it and stores the approved output.

2.2 The Internet of Things

The “Internet of things” (IoT) [\[2\]](#) is becoming an increasingly growing topic of conversation both in the workplace and outside of it. It’s a concept that not only has the potential to impact how we live but also how we work. But what exactly is the “Internet of things” and what impact is it going to have on you, if any? There are a lot of complexities around the “Internet of things” but I want to stick to the basics. Lots of technical and policy-related conversations are being had but many people are still just trying to grasp the foundation of what the heck these conversations are about.

Let’s start with understanding a few things.

Broadband Internet is become more widely available, the cost of connecting is decreasing, more devices are being created with Wi-Fi capabilities and sensors built into them, technology costs are going down, and smartphone penetration is sky-rocketing. All of these things are creating a “perfect storm” for the IoT.

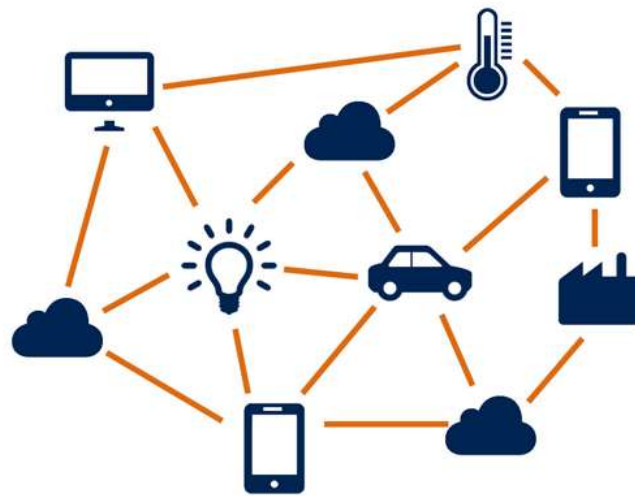


Figure (2.3): IOT Map

Internet of Things

Simply put, this is the concept of basically connecting any device with an on and off switch to the Internet (and/or to each other). This includes everything from cellphones, coffee makers, washing machines, headphones, lamps, wearable devices and almost anything else you can think of. This also applies to components of machines, for example a jet engine of an airplane or the drill of an oil rig. As I mentioned, if it has an on and off switch then chances are it can be a part of the IoT. The analyst firm Gartner says that by 2020 there will be over 26 billion connected devices... That's a lot of connections (some even estimate this number to be much higher, over 100 billion). The IoT is a giant network of connected "things" (which also includes people). The relationship will be between people-people, people-things, and things-things.

Impact

The new rule for the future is going to be, "Anything that can be connected, will be connected." But why on earth would you want so many connected devices talking to each other? There are many examples for what this might look like or what the potential value might be. Say for example you are on your way to a meeting; your car could have access to your calendar and already know the best route to take. If the traffic is heavy your car might send a text to the other party notifying them that you will be late. What if your alarm clock wakes up you at 6 a.m. and then notifies your coffee maker to start brewing coffee for you? What if your office equipment knew when it was running low on supplies and automatically re-ordered more? What if the wearable device you used in the workplace could tell you when and where you were most active and productive and shared that information with other devices that you used while working?

Chapter (3): Microcontrollers

A microcontroller [\[3\]](#) is a compact microcomputer designed to govern the operation of embedded systems in motor vehicles, robots, office machines, complex medical devices, mobile radio transceivers, vending machines, home appliances, and various other devices. A typical microcontroller includes a processor, memory, and peripherals.

The simplest microcontrollers facilitate the operation of the electromechanical systems found in everyday convenience items. Originally, such use was confined to large machines such as furnaces and automobile engines to optimize efficiency and performance. In recent years, microcontrollers have found their way into common items such as ovens, refrigerators, toasters, clock radios, and lawn watering systems. Microcomputers are also common in office machines such as photocopiers, scanners, fax machines, and printers.

The most sophisticated microcontrollers perform critical functions in aircraft, spacecraft, ocean-going vessels, life-support systems, and robots of all kinds. Medical technology offers especially promising future roles. For example, a microcontroller might regulate the operation of an artificial heart, artificial kidney, or other artificial body organ. Microcomputers can also function with prosthetic devices (artificial limbs). A few medical-science futurists have suggested that mute patients might someday be able, in effect, to speak out loud by thinking of the words they want to utter, while a microcontroller governs the production of audio signals to drive an amplifier and loudspeaker.

Microcomputers enjoy immense popularity among electronics hobbyists and experimenters. Perhaps the most widely known and used of these devices belong to the PIC family, manufactured by Microchip Technology, Inc. of Chandler, Arizona. All devices in the PIC family come with a wide variety of development tools, are easy to find, remain relatively inexpensive, and have excellent documentation.

3.1 Raspberry Pi

The Raspberry Pi [\[4\]](#) is a series of credit card-sized single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and developing countries.



Figure (3.1): Raspberry Pi 3

3.1.1 Overview

Several generations of Raspberry Pis have been released. The first generation (Raspberry Pi 1 Model B) was released in February 2012. It was followed by a simpler and inexpensive model A. In 2014 the foundation released a board with an improved design in Raspberry Pi 1 Model B+. The model laid the current "mainline" form-factor. Improved A+ and B+ models were released a year later. A cut down "compute" model was released in April 2014, and a Raspberry Pi Zero with smaller size and limited input/output (I/O) and general-purpose input/output (GPIO) abilities was released in November 2015 for US\$5. The Raspberry Pi 2 which added more RAM was released in February 2015. Raspberry Pi 3 Model B released in February 2016 is bundled with on-board WiFi and Bluetooth. As of 2016, Raspberry Pi 3 Model B is the newest mainline Raspberry Pi. These boards are priced between US \$20–35.

Chapter (3): Microcontrollers

All models feature a Broadcom system on a chip (SoC), which includes an ARM compatible central processing unit (CPU) and an on chip graphics processing unit (GPU, a VideoCore IV). CPU speed ranges from 700 MHz to 1.2 GHz for the Pi 3 and on board memory range from 256 MB to 1 GB RAM. Secure Digital (SD) cards are used to store the operating system and program memory in either the SDHC or MicroSDHC sizes. Most boards have between one and four USB slots, HDMI and composite video output, and a 3.5 mm phone jack for audio. Lower level output is provided by a number of GPIO pins which support common protocols like I²C. The B-models have an 8P8C Ethernet port and the Pi 3 has on board Wi-Fi 802.11n and Bluetooth.

The Foundation provides Raspbian, a Debian-based Linux distribution for download, as well as third party Ubuntu, Windows 10 IOT Core, RISC OS, and specialized media center distributions. It promotes Python and Scratch as the main programming language, with support for many other languages. The default firmware is closed source, while an unofficial open source is available.

In February 2016, the Raspberry Pi Foundation announced that they had sold eight million devices, making it the best-selling UK personal computer, ahead of the Amstrad PCW. Sales reached ten million in September 2016.

3.1.2 Hardware

The Raspberry Pi hardware has evolved through several versions that feature variations in memory capacity and peripheral-device support.

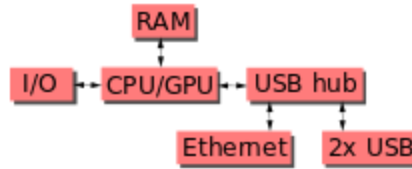


Figure (3.2): raspberry pi 3 hardware

This block diagram depicts Models A, B, A+, and B+. Model A, A+, and the Pi Zero lack the Ethernet and USB hub components. The Ethernet adapter is internally connected to an additional USB port. In Model A, A+, and the Pi Zero, the USB port is connected directly to the system on a chip (SoC). On the Pi 1 Model B+ and later models the USB/Ethernet chip contains a five-point USB hub, of which four ports are available, while the Pi 1 Model B only provides two. On the Pi Zero, the USB port is also connected directly to the SoC, but it uses a micro USB (OTG) port.

3.1.3 Processor

The Raspberry Pi 2 uses a 32-bit 900 MHz quad-core ARM Cortex-A7 processor.

The Broadcom BCM2835 SoC used in the first generation Raspberry Pi is somewhat equivalent to the chip used in first generation smartphones (its CPU is an older ARMv6 architecture), which includes a 700 MHz ARM1176JZF-S processor, VideoCore IV graphics processing unit (GPU), and RAM. It has a level 1 (L1) cache of 16 KB and a level 2 (L2) cache of 128 KB. The level 2 cache is used primarily by the GPU. The SoC is stacked underneath the RAM chip, so only its edge is visible.

The Raspberry Pi 2 uses a Broadcom BCM2836 SoC with a 900 MHz 32-bit quad-core ARM Cortex-A7 processor (as do many current smartphones), with 256 KB shared L2 cache.

The Raspberry Pi 3 uses a Broadcom BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor, with 512 KB shared L2 cache.

3.1.4 Performance

While operating at 700 MHz by default, the first generation Raspberry Pi provided a real-world performance roughly equivalent to 0.041 GFLOPS. On the CPU level the performance is similar to a 300 MHz Pentium II of 1997–99. The GPU provides 1 Gpixel/s or 1.5 Gtexel/s of graphics processing or 24 GFLOPS of general purpose computing performance. The graphical capability of the Raspberry Pi is roughly equivalent to the performance of the Xbox of 2001.

The LINPACK single node compute benchmark results in a mean single precision performance of 0.065 GFLOPS and a mean double precision performance of 0.041 GFLOPS for one Raspberry Pi Model-B board. A cluster of 64 Raspberry Pi Model B computers, labeled "Iridis-pi", achieved a LINPACK HPL suite result of 1.14 GFLOPS (n=10240) at 216 watts for c. US\$4000.

Raspberry Pi 2 includes a quad-core Cortex-A7 CPU running at 900 MHz and 1 GB RAM. It is described as 4–6 times more powerful than its predecessor. The GPU is identical to the original. In parallelized benchmarks, the Raspberry Pi 2 could be up to 14 times faster than a Raspberry Pi 1 Model B+.

The Raspberry Pi 3, with a quad-core Cortex-A53 processor, is described as 10 times the performance of a Raspberry Pi 1. This was suggested to be highly dependent upon task threading and instruction set use. Benchmarks showed the Raspberry Pi 3 to be approximately 80% faster than the Raspberry Pi 2 in parallelized tasks.

3.1.4 Overclocking

The first generation Raspberry Pi chip operated at 700 MHz by default, and did not become hot enough to need a heat sink or special cooling unless the chip was overclocked. The Raspberry Pi 2 runs at 900 MHz by default; it also does not become hot enough to need a heatsink or special cooling, although overclocking may heat up the SoC more than usual.

Chapter (3): Microcontrollers

Most Raspberry Pi chips could be overclocked to 800 MHz, and some to 1000 MHz. There are reports the Raspberry Pi 2 can be similarly overclocked, in extreme cases, even to 1500 MHz (discarding all safety features and over-voltage limitations). In the Raspbian Linux distro the overclocking options on boot can be done by a software command running "sudo raspi-config" without voiding the warranty. In those cases, the Pi automatically shuts the overclocking down if the chip reaches 85 °C (185 °F), but it is possible to override automatic over-voltage and overclocking settings (voiding the warranty); an appropriately sized heatsink is needed to protect the chip from serious overheating.

Newer versions of the firmware contain the option to choose between five overclock ("turbo") presets that when used, attempt to maximize the performance of the SoC without impairing the lifetime of the board. This is done by monitoring the core temperature of the chip, the CPU load, and dynamically adjusting clock speeds and the core voltage. When the demand is low on the CPU or it is running too hot the performance is throttled, but if the CPU has much to do and the chip's temperature is acceptable, performance is temporarily increased with clock speeds of up to 1 GHz depending on the individual board and on which of the turbo settings is used.

The seven overclock presets are:

none; 700 MHz ARM, 250 MHz core, 400 MHz SDRAM, 0 overvolt,

modest; 800 MHz ARM, 250 MHz core, 400 MHz SDRAM, 0 overvolt,

medium; 900 MHz ARM, 250 MHz core, 450 MHz SDRAM, 2 overvolt,

high; 950 MHz ARM, 250 MHz core, 450 MHz SDRAM, 6 overvolt,

turbo; 1000 MHz ARM, 500 MHz core, 600 MHz SDRAM, 6 overvolt,

Pi2; 1000 MHz ARM, 500 MHz core, 500 MHz SDRAM, 2 overvolt,

Pi3; 1100 MHz ARM, 550 MHz core, 500 MHz SDRAM, 6 overvolt. In system information CPU speed will appear as 1200 MHz. When in idle speed lowers to 600 MHz.

In the highest (turbo) preset the SDRAM clock was originally 500 MHz, but this was later changed to 600 MHz because 500 MHz sometimes causes SD card corruption. Simultaneously in high mode the core clock speed was lowered from 450 to 250 MHz, and in medium mode from 333 to 250 MHz.

The Raspberry Pi Zero runs at 1 GHz.

3.1.5 RAM

On the older beta Model B boards, 128 MB was allocated by default to the GPU, leaving 128 MB for the CPU. On the first 256 MB release Model B (and Model A), three different splits were possible. The default split was 192 MB (RAM for CPU), which should be sufficient for standalone 1080p video decoding, or for simple 3D, but probably not for both together. 224 MB was for Linux only, with only a 1080p framebuffer, and was likely to fail for any video or 3D. 128 MB was for heavy 3D, possibly also with video decoding (e.g. XBMC). Comparatively the Nokia 701 uses 128 MB for the Broadcom VideoCore IV. For the new Model B with 512 MB RAM initially there were new standard memory split files released (arm256_start.elf, arm384_start.elf, arm496_start.elf) for 256 MB, 384 MB and 496 MB CPU RAM (and 256 MB, 128 MB and 16 MB video RAM). But a week or so later the RPF released a new version of start.elf that could read a new entry in config.txt (gpu_mem=xx) and could dynamically assign an amount of RAM (from 16 to 256 MB in 8 MB steps) to the GPU, so the older method of memory splits became obsolete, and a single start.elf worked the same for 256 and 512 MB Raspberry Pis.

The Raspberry Pi 2 and the Raspberry Pi 3 have 1 GB of RAM. The Raspberry Pi Zero has 512 MB of RAM.

3.1.6 Networking

The Model A, A+ and Pi Zero have no Ethernet circuitry and are commonly connected to a network using an external user-supplied USB Ethernet or Wi-Fi adapter. On the Model B and B+ the Ethernet port is provided by a built-in USB Ethernet adapter using the SMSC LAN9514 chip. The Raspberry Pi 3 is equipped with 2.4 GHz WiFi 802.11n (150 Mbit/s) and Bluetooth 4.1 (24 Mbit/s) in addition to the 10/100 Ethernet port.

3.1.7 Peripherals

The current Model B boards incorporate four USB ports for connecting peripherals.

The Raspberry Pi may be operated with any generic USB computer keyboard and mouse.

3.1.8 Video

The early Raspberry Pi 1 Model A, with an HDMI port and a standard RCA composite video port for older displays.

The video controller can emit standard modern TV resolutions, such as HD and Full HD, and higher or lower monitor resolutions and older standard CRT TV resolutions. As shipped (i.e., without custom overclocking) it can emit these: 640×350 EGA; 640×480 VGA; 800×600 SVGA; 1024×768 XGA; 1280×720 720p HDTV; 1280×768 WXGA variant; 1280×800 WXGA variant; 1280×1024 SXGA; 1366×768 WXGA variant; 1400×1050 SXGA+; 1600×1200 UXGA; 1680×1050 WXGA+; 1920×1080 1080p HDTV; 1920×1200 WUXGA.

Higher resolutions, such as, up to 2048×1152, may work or even 3840×2160 at 15 Hz (too low a framerate for convincing video). Note also that allowing the highest resolutions does not imply that the GPU can decode video formats at those; in fact, the Pis are known to not work reliably for H.265 (at those high resolution, at least), commonly used for very high resolutions (most formats, commonly used, up to full HD, do work).

Although the Raspberry Pi 3 does not have H.265 decoding hardware, the CPU, more powerful than its predecessors, is potentially able to decode H.265-encoded videos in software. The Open Source Media Center (OSMC) project said in February 2016:

The new BCM2837 based on 64-bit ARMv8 architecture is backwards compatible with the Raspberry Pi 2 as well as the original. While the new CPU is 64-bit, the Pi retains the original VideoCore IV GPU which has a 32-bit design. It will be a few months before work is done to establish 64-bit pointer interfacing from the kernel and userland on the ARM to the 32-bit GPU. As such, for the time being, we will be offering a single Raspberry Pi image for Raspberry Pi 2 and the new Raspberry Pi 3. Only when 64-bit support is ready, and beneficial to OSMC users

The new quad core CPU will bring smoother GUI performance. There have also been recent improvements to H265 decoding. While not hardware accelerated on the Raspberry Pi, the new CPU will enable more H265 content to be played back on the Raspberry Pi than before.

— Raspberry Pi 3 announced with OSMC support

The Pi 3's GPU has higher clock frequencies—300 MHz and 400 MHz for different parts—than previous versions' 250 MHz.

The Raspberry Pi can also generate 576i and 480i composite video signals, as used on old-style (CRT) TV screens through standard connectors—either RCA or 3.5mm phone connector depending on models. The television signal standards supported are PAL-BGHID, PAL-M, PAL-N, NTSC and NTSC-J.

3.1.9 Real-time clock

The Raspberry Pi does not have a built-in real-time clock, and does not "know" the time of day. As a workaround, a program running on the Raspberry Pi can get the time from a network time server or user input at boot time, thus knowing the time while powered on. A real-time hardware clock with battery backup, such as the DS1307, which is fully binary coded, may be added (often via the I²C interface).

3.2 Arduino

Arduino [\[5\]](#) is an open-source project that created microcontroller-based kits for building digital devices and interactive objects that can sense and control physical devices.

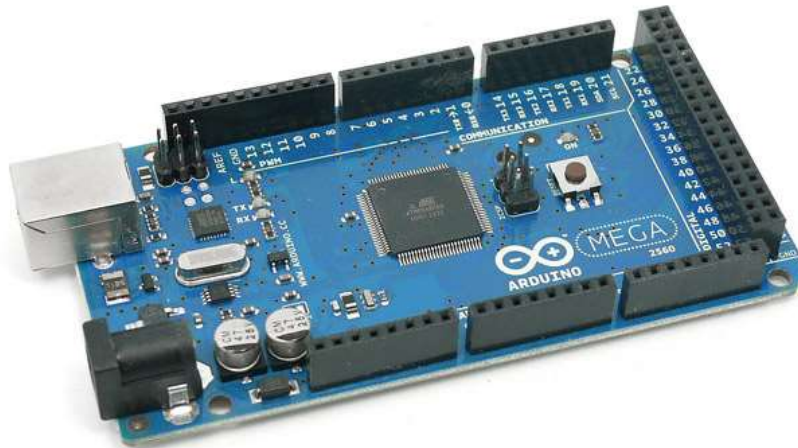


Figure (3.3): Arduino mega

3.2.1 Overview

Arduino is an open source hardware and software project, created with a simple aim in mind, to be as simple as possible. Arduino is not some hardware you should be afraid of. Arduino comes in a variety of flavors and sizes. It is used by artists, hackers, hobbyists, and professionals to easily design, prototype and experiment with electronics. Use it as brains for your robot, to build a new digital music instrument, or to make your house plant tweet you when it's dry. An Arduino contains a microchip, which is a very small computer that you can program. You can attach sensors to it so that it can measure conditions (like how much light there is in the room). It can control how other objects react to those conditions (room gets dark. LED turns on).

Chapter (3): Microcontrollers

The project is based on microcontroller board designs, produced by several vendors, using various microcontrollers. Microcontrollers use inputs and outputs like any computer. Inputs capture information from the user or the environment while outputs do something with the information that has been captured. A switch and a sensor could be a digital and an analog input respectively into the Arduino. Any object we want to turn on and off and control could be an output. It could be a motor or even a computer. These systems provide sets of digital and analog input/output (I/O) pins that can interface to various expansion boards (termed shields) and other circuits. The boards feature serial communication interfaces, including Universal Serial Bus (USB) on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino project provides an integrated development environment (IDE) based on a programming language named Processing, which also supports the languages C and C++. The Arduino language is very similar to C. It's almost the same language but Arduino provides us with several libraries to make things a bit easier.

The first Arduino was introduced in 2005, based on 8-bit Atmel AVR, aiming to provide a low cost, easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

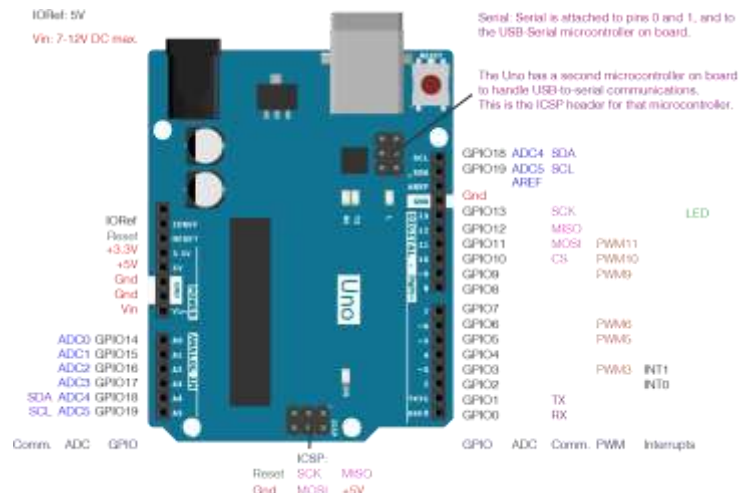
Arduino comes in a variety of different boards. Arduino boards are available commercially in pre-assembled form, or as do-it-yourself kits. The hardware design specifications are openly available, allowing the Arduino boards to be produced by anyone. In mid-2011, it was estimated that over 300,000 official Arduinos had been commercially produced, and in 2013 that 700,000 official boards were in users' hands.

3.2.2 Hardware

An Arduino board consists of an Atmel 8-, 16- or 32-bit AVR microcontroller (although since 2015 other makers' microcontrollers have been used) with complementary components that facilitate programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which let users connect the CPU board to a variety of interchangeable add-on modules termed shields. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an I²C serial bus—so many shields can be stacked and used in parallel. Before 2015, Official Arduinos had used the Atmel megaAVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. In 2015, units by other producers were added. A handful of other processors have also been used by Arduino compatible devices. Most boards include a 5 V linear regulator and a 16 MHz crystal oscillator (or ceramic resonator in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external chip programmer. This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer. Currently, optiboot bootloader is the default bootloader installed on Arduino UNO.

At a conceptual level, when using the Arduino integrated development environment, all boards are programmed over a serial connection. Its implementation varies with the hardware version. Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and transistor–transistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. Some boards, such as later-model Uno boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods, when used with traditional microcontroller tools instead of the Arduino IDE, standard AVR in-system programming (ISP) programming is used.

Chapter (3): Microcontrollers



An official Arduino Uno with descriptions of the I/O locations

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, [a]Duemilanove, [b] and current Uno[c] provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.1-inch (2.54 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into solderless breadboards.

Many Arduino-compatible and Arduino-derived boards exist. Some are functionally equivalent to an Arduino and can be used interchangeably. Many enhance the basic Arduino by adding output drivers, often for use in school-level education, to simplify making buggies and small robots. Others are electrically equivalent but change the form factor, sometimes retaining compatibility with shields, sometimes not. Some variants use different processors, of varying compatibility.

3.2.3 Shields

Arduino and Arduino-compatible boards use printed circuit expansion boards called shields, which plug into the normally supplied Arduino pin headers. Shields can provide motor controls for 3D printing and other applications, Global Positioning System (GPS), Ethernet, liquid crystal display (LCD), or breadboarding (prototyping). Several shields can also be made do it yourself (DIY).

3.2.4 Software

Arduino IDE



Figure (3.4): Screenshot of the Arduino IDE showing the Blink Program

| | |
|--------------------------|---|
| Developer(s): | Arduino |
| Stable release: | 1.6.12 / 21 September 2016 |
| Repository: | github.com/arduino/Arduino |
| Written in: | Java, C and C++ |
| Operating system: | Windows, OS X, Linux |
| Platform: | IA-32, x86-64, ARM |
| Type: | Integrated development environment |
| License: | LGPL or GPL license |
| Website: | arduino.cc |

3.2.5 Arduino IDE

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It was created for people with no profound knowledge of electronics. It includes a code editor with features such as syntax highlighting, brace matching, cutting/pasting text, searching/replacing text and automatic indentation, and provides simple one-click mechanism to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a series of menus.

A program written with the IDE for Arduino is called a "sketch". Sketches are saved on the development computer as files with the file extension. `.ino`. Arduino Software (IDE) prior to 1.0 saved sketches with the extension. `.pde`.

The Arduino IDE supports the languages C and C++ using special rules to organize code. The Arduino IDE supplies a software library called Wiring from the Wiring project, which provides many common input and output procedures. A typical Arduino C/C++ sketch consist of two functions that are compiled and linked with a program stub `main()` into an executable cyclic executive program:

`setup()`: this function is called once when a sketch starts after power-up or reset. It is used to initialize variables, pin modes, start using libraries, etc.

`loop()`: after `setup()` is called, this function is called repeatedly until the board powers off. It actively controls the Arduino board and allows the program to change or respond.

After compiling and linking with the GNU toolchain, also included with the IDE distribution, the Arduino IDE employs the program `avrdude` to convert the executable code into a text file in hexadecimal coding that is loaded into the Arduino board by a loader program in the board's firmware.

Chapter (4): Electronics-components

4.1 Diode:

4.1.1 Introduction

Once you graduate from the simple, passive components that are resistors, capacitors, and inductors, it's time to step on up to the wonderful world of semiconductors. One of the most widely used semiconductor components is the diode [\[6\]](#).



Figure (4.1): Diode

4.1.2 Ideal Diodes

The key function of an ideal diode is to control the direction of current-flow. Current passing through a diode can only go in one direction, called the forward direction. Current trying to flow the reverse direction is blocked. They're like the one-way valve of electronics.

If the voltage across a diode is negative, no current can flow*, and the ideal diode looks like an open circuit. In such a situation, the diode is said to be off or reverse biased.

As long as the voltage across the diode isn't negative, it'll "turn on" and conduct current. Ideally* a diode would act like a short circuit (0V across it) if it was conducting current. When a diode is conducting current it's forward biased (electronics jargon for "on").



The current-voltage relationship of an ideal diode. Any negative voltage produces zero current – an open circuit. As long as the voltage is non-negative the diode looks like a short circuit.

Figure (4.2): The current-voltage relationship of an ideal diode

4.1.3 Types of Diodes

Normal Diodes

Standard signal diodes are among the most basic, average, no-frills members of the diode family. They usually have a medium-high forward voltage drop and a low maximum current rating. A common example of a signal diode is the 1N4148. Very general purpose, it's got a typical forward voltage drop of 0.72V and a 300mA maximum forward current rating.

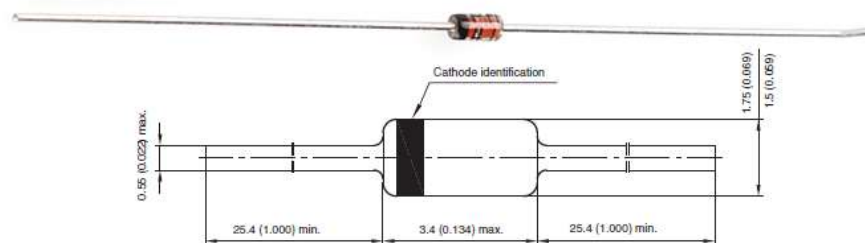


Figure (4.3): diode cathode identification

A small-signal diode, the 1N4148. Notice the black circle around the diode that marks which of the terminals is the cathode.

A rectifier or power diode is a standard diode with a much higher maximum current rating. This higher current rating usually comes at the cost of a larger forward voltage. The 1N4001, for example, has a current rating of 1A and a forward voltage of 1.1V.



A 1N4001 PTH diode. This time a gray band indicates which pin is the cathode.

And, of course, most diode types come in surface-mount varieties as well. You'll notice that every diode has some way (no matter how tiny or hard to see) to indicate which of the two pins is the cathode.



Figure (4.4): diode types

Light-Emitting Diodes (LEDs!)

The flashiest member of the diode family must be the light-emitting diode (LED). These diodes quite literally light up when a positive voltage is applied.

Schottky Diodes

Another very common diode is the Schottky diode. The semiconductor composition of a Schottky diode is slightly different from a normal diode, and this results in a much smaller forward voltage drop, which is usually between 0.15V and 0.45V. They'll still have a very large breakdown voltage though.

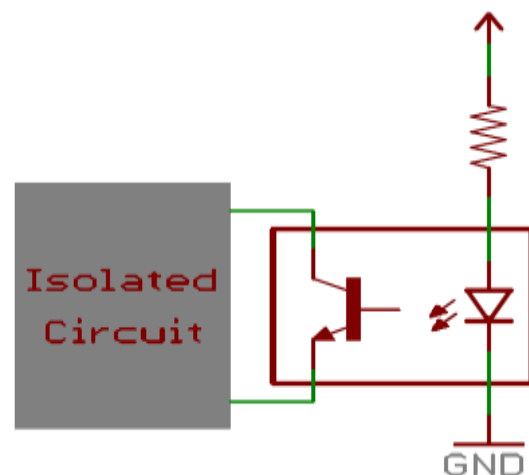


Figure (4.5): Schottky diodes isolated circuit

Schottky diodes are especially useful in limiting losses, when every last bit of voltage must be spared. They're unique enough to get a circuit symbol of their own, with a couple bends on the end of the cathode-line.

Zener Diodes

Zener diodes are the weird outcast of the diode family. They're usually used to intentionally conduct reverse current. Zener's are designed to have a very precise breakdown voltage, called the zener breakdown or zener voltage. When enough current runs in reverse through the zener, the voltage drop across it will hold steady at the breakdown voltage.

Taking advantage of their breakdown property, Zener diodes are often used to create a known reference voltage at exactly their Zener voltage. They can be used as a voltage regulator for small loads, but they're not really made to regulate voltage to circuits that will pull significant amounts of current.

Zeners are special enough to get their own circuit symbol, with wavy ends on the cathode-line. The symbol might even define what, exactly, the diode's zener voltage is. Here's a 3.3V zener diode acting to create a solid 3.3V voltage reference:

4.2 Light-emitting diode (LED):

A light-emitting diode (LED) [\[7\]](#) is a semiconductor device that emits visible light when an electric current passes through it. The light is not particularly bright, but in most LEDs it is monochromatic, occurring at a single wavelength. The output from an LED can range from red (at a wavelength of approximately 700 nanometers) to blue-violet (about 400 nanometers). Some LEDs emit infrared (IR) energy (830 nanometers or longer); such a device is known as an infrared-emitting diode (IRED).

An LED or IRED consists of two elements of processed material called P-type semiconductors and N-type semiconductors. These two elements are placed in direct contact, forming a region called the P-N junction. In this respect, the LED or IRED resembles most other diode types, but there are important differences. The LED or IRED has a transparent package, allowing visible or IR energy to pass through. Also, the LED or IRED has a large PN-junction area whose shape is tailored to the application.

Benefits of LEDs and IREDs, compared with incandescent and fluorescent illuminating devices, include:

Low power requirement: Most types can be operated with battery power supplies.

High efficiency: Most of the power supplied to an LED or IRED is converted into radiation in the desired form, with minimal heat production.

Long life: When properly installed, an LED or IRED can function for decades.

Typical applications include:

Indicator lights: These can be two-state (i.e., on/off), bar-graph, or alphabetic-numeric readouts.

Chapter(4):
Basic electronics

LCD panel backlighting: Specialized white LEDs are used in flat-panel computer displays.

Fiber optic data transmission: Ease of modulation allows wide communications bandwidth with minimal noise, resulting in high speed and accuracy.

Remote control: Most home-entertainment "remotes" use IREDs to transmit data to the main unit.

4.3 Resistors:

A resistor [\[8\]](#) is an electrical component that limits or regulates the flow of electrical current in an electronic circuit. Resistors can also be used to provide a specific voltage for an active device such as a transistor.

All other factors being equal, in a direct-current (DC) circuit, the current through a resistor is inversely proportional to its resistance, and directly proportional to the voltage across it. This is the well-known Ohm's Law. In alternating-current (AC) circuits, this rule also applies as long as the resistor does not contain inductance or capacitance.

Resistors can be fabricated in a variety of ways. The most common type in electronic devices and systems is the carbon-composition resistor. Fine granulated carbon (graphite) is mixed with clay and hardened. The resistance depends on the proportion of carbon to clay; the higher this ratio, the lower the resistance.

Another type of resistor is made from winding Nichrome or similar wire on an insulating form. This component, called a wirewound resistor, is able to handle higher currents than a carbon-composition resistor of the same physical size. However, because the wire is wound into a coil, the component acts as some inductors as well as exhibiting resistance. This does not affect performance in DC circuits, but can have an adverse effect in AC circuits because inductance renders the device sensitive to changes in frequency.

4.4 Ohm's Law:

Combining the elements of voltage, current, and resistance, Ohm developed the formula:

$$V = I \cdot R$$

Where

- V = Voltage in volts
- I = Current in amps
- R = Resistance in ohms

This is called Ohm's law [\[9\]](#). Let's say, for example, that we have a circuit with the potential of 1 volt, a current of 1 amp, and resistance of 1 ohm. Using Ohm's Law, we can say:

$$1V = 1A \cdot 1\Omega$$

Let's say this represents our tank with a wide hose. The amount of water in the tank is defined as 1 volt and the "narrowness" (resistance to flow) of the hose is defined as 1 ohm. Using Ohms Law, this gives us a flow (current) of 1 amp.

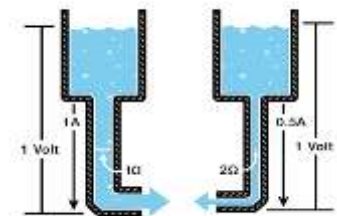
Using this analogy, let's now look at the tank with the narrow hose. Because the hose is narrower, its resistance to flow is higher. Let's define this resistance as 2 ohms. The amount of water in the tank is the same as the other tank, so, using Ohm's Law, our equation for the tank with the narrow hose is

$$1V = ?A \cdot 2\Omega$$

But what is the current? Because the resistance is greater, and the voltage is the same, this gives us a current value of 0.5 amps:

$$1V = 0.5A \cdot 2\Omega$$

So, the current is lower in the tank with higher resistance.



4.5 Capacitor:

Capacitor [\[10\]](#) is an electronic component that stores electric charge. The capacitor is made of 2 close conductors (usually plates) that are separated by a dielectric material. The plates accumulate electric charge when connected to power source. One plate accumulates positive charge and the other plate accumulates negative charge.

The capacitance is the amount of electric charge that is stored in the capacitor at voltage of 1 Volt.

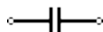
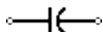

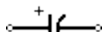
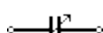
The capacitance is measured in units of Farad (F).

The capacitor disconnects current in direct current (DC) circuits and short circuit in alternating current (AC) circuits.

Capacitor Types



4.5.1 Capacitor symbols

| | | |
|---------------------|---|---|
| Capacitor |  |  |
| Polarized Capacitor |  |  |
| Variable Capacitor |  | |

4.5.2 Capacitance

The capacitance (C) of the capacitor is equal to the electric charge (Q) divided by the voltage (V):

$$C = \frac{Q}{V}$$

C is the capacitance in farad (F)

Q is the electric charge in coulombs (C), that is stored on the capacitor

V is the voltage between the capacitor's plates in volts (V)

4.5.3 Capacitance of plates capacitor

The capacitance (C) of the plates capacitor is equal to the permittivity (ϵ) times the plate area (A) divided by the gap or distance between the plates (d):

$$C = \epsilon \times \frac{A}{d}$$

C is the capacitance of the capacitor, in farad (F).

ϵ is the permittivity of the capacitor's dielectric material, in farad per meter (F/m).

A is the area of the capacitor's plate in square meters (m²).

d is the distance between the capacitor's plates, in meters (m).

4.6 Transistor:

A transistor [\[11\]](#) is a device that regulates current or voltage flow and acts as a switch or gate for electronic signals. Transistors consist of three layers of a semiconductor material, each capable of carrying a current.

The transistor was invented by three scientists at the Bell Laboratories in 1947, and it rapidly replaced the vacuum tube as an electronic signal regulator. A transistor regulates current or voltage flow and acts as a switch or gate for electronic signals. A transistor consists of three layers of a semiconductor material, each capable of carrying a current. A semiconductor is a material such as germanium and silicon that conducts electricity in a "semi-enthusiastic" way. It's somewhere between a real conductor such as copper and an insulator (like the plastic wrapped around wires).

The semiconductor material is given special properties by a chemical process called doping. The doping results in a material that either adds extra electrons to the material (which is then called N-type for the extra negative charge carriers) or creates "holes" in the material's crystal structure (which is then called P-type because it results in more positive charge carriers). The transistor's three-layer structure contains an N-type semiconductor layer sandwiched between P-type layers (a PNP configuration) or a P-type layer between N-type layers (an NPN configuration).

A small change in the current or voltage at the inner semiconductor layer (which acts as the control electrode) produces a large, rapid change in the current passing through the entire component. The component can thus act as a switch, opening and closing an electronic gate many times per second. Today's computers use circuitry made with complementary metal oxide semiconductor (CMOS) technology. CMOS uses two complementary transistors per gate (one with N-type material; the other with P-type material). When one transistor is maintaining a logic state, it requires almost no power.

Transistors are the basic elements in integrated circuits (IC), which consist of very large numbers of transistors interconnected with circuitry and baked into a single silicon microchip.

4.7 Switch:

The "push-button" [\[12\]](#) has been utilized in calculators, push-button telephones, kitchen appliances, and various other mechanical and electronic devices, home and commercial. In industrial and commercial applications, push buttons can be connected together by a mechanical linkage so that the act of pushing one button causes the other button to be released. In this way, a stop button can "force" a start button to be released. This method of linkage is used in simple manual operations in which the machine or process have no electrical circuits for control.

Pushbuttons are often color-coded to associate them with their function so that the operator will not push the wrong button in error. Commonly used colors are red for stopping the machine or process and green for starting the machine or process.

Red pushbuttons can also have large heads (called mushroom heads) for easy operation and to facilitate the stopping of a machine. These pushbuttons are called emergency stop buttons and are mandated by the electrical code in many jurisdictions for increased safety. This large mushroom shape can also be found in buttons for use with operators who need to wear gloves for their work and could not actuate a regular flush-mounted push button. As an aid for operators and users in industrial or commercial applications, a pilot light is commonly added to draw the attention of the user and to provide feedback if the button is pushed. Typically, this light is included into the center of the pushbutton and a lens replaces the pushbutton hard center disk. The source of the energy to illuminate the light is not directly tied to the contacts on the back of the pushbutton but to the action the pushbutton controls. In this way a start button when pushed will cause the process or machine operation to be started and a secondary contact designed into the operation or process will close to turn on the pilot light and signify the action of pushing the button caused the resultant process or action to start.

In popular culture, the phrase "the button" (sometimes capitalized) refers to a (usually fictional) button that a military or government leader could press to launch nuclear weapons.

4.8 Relay:

A relay [\[13\]](#) is a simple electromechanical switch made up of an electromagnet and a set of contacts. Relays are found hidden in all sorts of devices. In fact, some of the first computers ever built used relays to implement Boolean gates.

4.8.1 Relay Construction

Relays are amazingly simple devices. There are four parts in every relay:

- Electromagnet
- Armature that can be attracted by the electromagnet
- Spring
- Set of electrical contacts

The following figure shows these four parts in action:

This content is not compatible on this device.

In this figure, you can see that a relay consists of two separate and completely independent circuits. The first is at the bottom and drives the electromagnet. In this circuit, a switch is controlling power to the electromagnet. When the switch is on, the electromagnet is on, and it attracts the armature (blue). The armature is acting as a switch in the second circuit. When the electromagnet is energized, the armature completes the second circuit and the light is on. When the electromagnet is not energized, the spring pulls the armature away and the circuit is not complete. In that case, the light is dark.

When you purchase relays, you generally have control over several variables:

- The voltage and current that is needed to activate the armature
- The maximum voltage and current that can run through the armature and the armature contacts
- The number of armatures (generally one or two)
- The number of contacts for the armature (generally one or two -- the relay shown here has two, one of which is unused)
- Whether the contact (if only one contact is provided) is normally open (NO) or normally closed (NC)

4.9 Breadboard:

A breadboard [\[14\]](#) is used to build and test circuits quickly before finalizing any circuit design. The breadboard has many holes into which circuit components like ICs and resistors can be inserted. A typical breadboard is shown below:

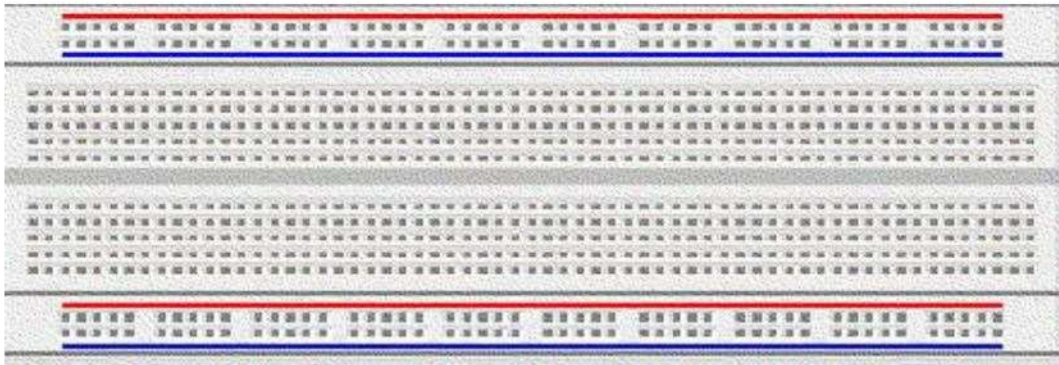
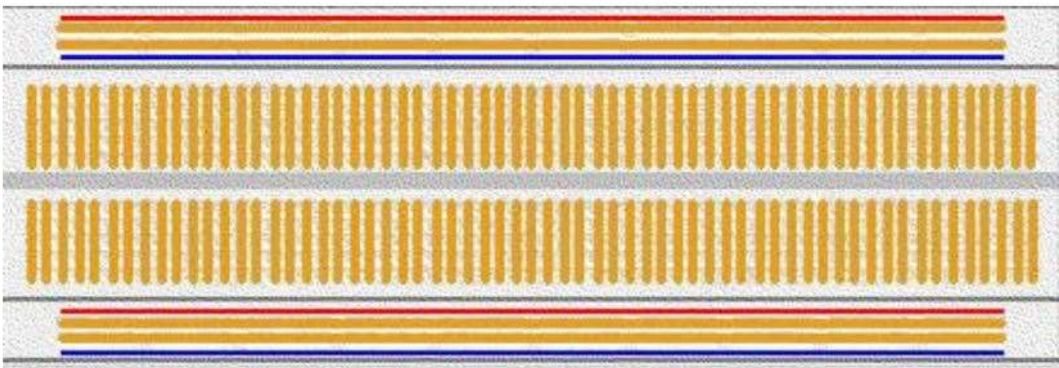


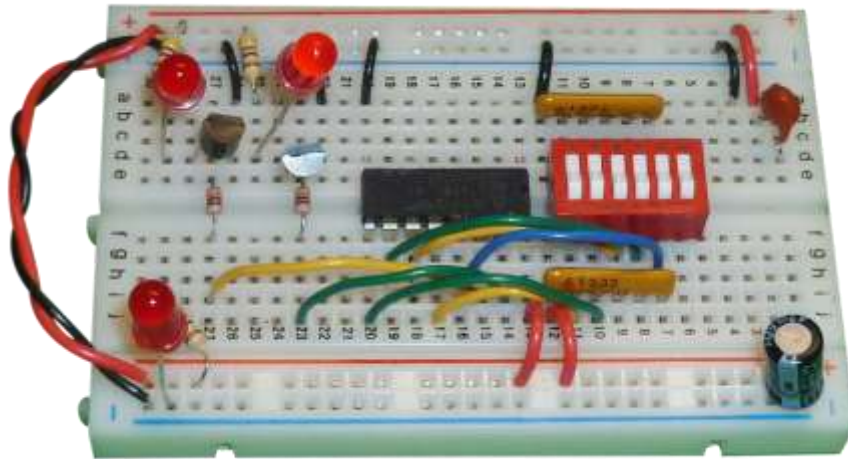
Figure (4.6): Breadboard

The bread board has strips of metal which run underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below



. Note that the top and bottom rows of holes are connected horizontally while the remaining holes are connected vertically. To use the bread board, the legs of components are placed in the holes. Each set of holes connected by a metal strip underneath forms a node. A node is a point in a circuit where two components are connected. Connections between different components are formed by putting their legs in a common node. The long top and bottom row of holes are usually used for power supply connections. The rest of the circuit is built by placing components and connecting them together with jumper wires. ICs are placed in the middle of the board so that half of the legs are on one side of the middle line and half on the other.

A completed circuit might look like the following.



Breadboarding tips: It is important to breadboard a circuit neatly and systematically, so that one can debug it and get it running easily and quickly. It also helps when someone else needs to understand and inspect the circuit. Here are some tips: 1. always use the side-lines for power supply connections. Power the chips from the side-lines and not directly from the power supply. 2. Use black wires for ground connections (0V), and red for other power connections. 3. Keep the jumper wires on the board flat, so that the board does not look cluttered. 4. Route jumper wires around the chips and not over the chips. This makes changing the chips when needed easier. 5. You could trim the legs of components like resistors, transistors and LEDs, so that they fit in snugly and do not get pulled out by accident.

Chapter (5): Sensors

5.1 Pyroelectric ("Passive") InfraRed sensors:

A passive infrared sensor (PIR sensor) [\[15\]](#) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors by house pets. Experimentation is key!

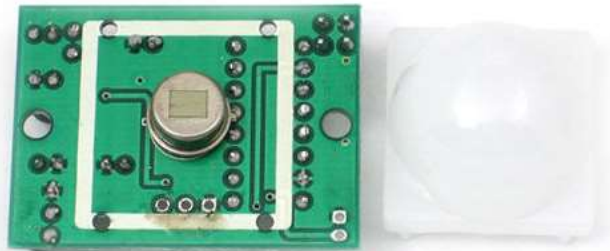


Figure (5.1): PIR Sensor

5.1.1 Operation

An individual PIR sensor detects changes in the amount of infrared radiation impinging upon it, which varies depending on the temperature and surface characteristics of the objects in front of the sensor. When an object, such as a human, passes in front of the background, such as a wall, the temperature at that point in the sensor's field of view will rise from room temperature to body temperature, and then back again. The sensor converts the resulting change in the incoming infrared radiation into a change in the output voltage, and this triggers the detection. Objects of similar temperature but different surface characteristics may also have a different infrared emission pattern, and thus moving them with respect to the background may trigger the detector as well.

Chapter (5):
Sensors

PIRs come in many configurations for a wide variety of applications. The most common models have numerous Fresnel lenses or mirror segments, an effective range of about ten meters (thirty feet), and a field of view less than 180 degrees. Models with wider fields of view, including 360 degrees, are available—typically designed to mount on a ceiling. Some larger PIRs are made with single segment mirrors and can sense changes in infrared energy over one hundred feet away from the PIR. There are also PIRs designed with reversible orientation mirrors which allow either broad coverage (110° wide) or very narrow "curtain" coverage, or with individually selectable segments to "shape" the coverage.

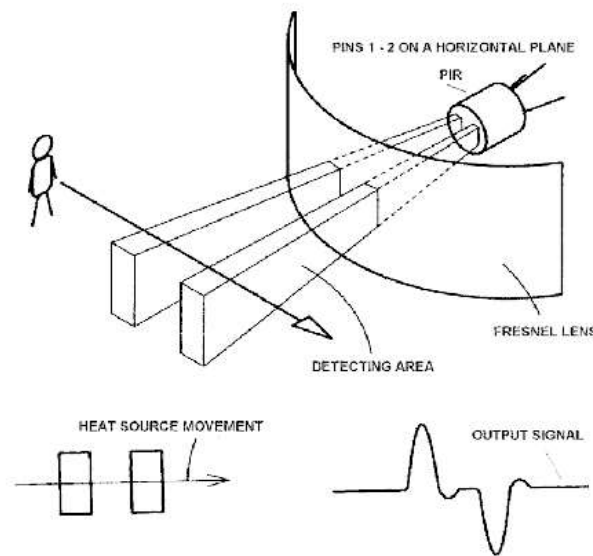


Figure (5.2): PIR Detection

5.2 Photoresistor



Figure (5.3): Photoresistor

A photoresistor (or light-dependent resistor, LDR, or photocell) [\[16\]](#) is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity; in other words, it exhibits photoconductivity. A photoresistor can be applied in light-sensitive detector circuits, and light- and dark-activated switching circuits.

A photoresistor is made of a high resistance semiconductor. In the dark, a photoresistor can have a resistance as high as several megohms ($M\Omega$), while in the light, a photoresistor can have a resistance as low as a few hundred ohms. If incident light on a photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electrons (and their whole partners) conduct electricity, thereby lowering resistance. The resistance range and sensitivity of a photoresistor can substantially differ among dissimilar devices. Moreover, unique photoresistors may react substantially differently to photons within certain wavelength bands.

A photoelectric device can be either intrinsic or extrinsic. An intrinsic semiconductor has its own charge carriers and is not an efficient semiconductor, for example, silicon. In intrinsic devices the only available electrons are in the valence band, and hence the photon must have enough energy to excite the electron across the entire bandgap. Extrinsic devices have impurities, also called dopants, and added whose ground state energy is closer to the conduction band; since the electrons do not have as far to jump, lower energy photons (that is, longer wavelengths and lower frequencies) are sufficient to trigger the device. If a sample of silicon has some of its atoms replaced by phosphorus atoms (impurities), there will be extra electrons available for conduction.

5.2.1 Design considerations

Photoresistors are less light-sensitive devices than photodiodes or phototransistors: the two latter components are true semiconductor devices, while a photoresistor is a passive component and does not have a PN-junction. The photoresistivity of any photoresistor may vary widely depending on ambient temperature, making them unsuitable for applications requiring precise measurement of or sensitivity to light.

Photoresistors also exhibit a certain degree of latency between exposure to light and the subsequent decrease in resistance, usually around 10 milliseconds. The lag time when going from lit to dark environments is even greater, often as long as one second. This property makes them unsuitable for sensing rapidly flashing lights, but is sometimes used to smooth the response of audio signal compression.

5.3 Ultrasonic transducer



Figure (5.4): Ultrasonic transducer

Ultrasonic transducers [\[17\]](#) are transducers that convert ultrasound waves to electrical signals or vice versa. Those that both transmit and receive may also be called ultrasound transceivers; many ultrasound sensors besides being sensors are indeed transceivers because they can both sense and transmit. These devices work on a principle similar to that of transducers used in radar and sonar systems, which evaluate attributes of a target by interpreting the echoes from radio or sound waves, respectively. Active ultrasonic sensors generate high-frequency sound waves and evaluate the echo which is received back by the sensor, measuring the time interval between sending the signal and receiving the echo to determine the distance to an object. Passive ultrasonic sensors are basically microphones that detect ultrasonic noise that is present under certain conditions, convert it to an electrical signal, and report it to a computer.

Ultrasonic probes and ultrasonic baths are used to apply sound energy to agitate particles in a wide range of laboratory applications; See Sonication.

5.3.1 Capabilities and limitations

This technology can be used for measuring wind speed and direction (anemometer), tank or channel fluid level, and speed through air or water. For measuring speed or direction, a device uses multiple detectors and calculates the speed from the relative distances to particulates in the air or water. To measure tank or channel level, the sensor measures the distance to the surface of the fluid. Further applications include: humidifiers, sonar, medical ultrasonography, burglar alarms, non-destructive testing and wireless charging.

Systems typically use a transducer which generates sound waves in the ultrasonic range, above 18 kHz, by turning electrical energy into sound, then upon receiving the echo turn the sound waves into electrical energy which can be measured and displayed.

The technology is limited by the shapes of surfaces and the density or consistency of the material. Foam, in particular, can distort surface level readings

This technology, as well, can detect approaching objects and track their positions.

5.3.2 Transducers

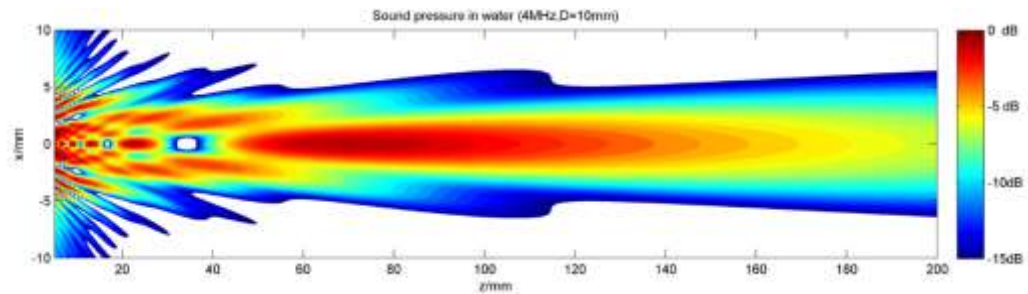


Figure (5.5): Sound field of a non-focusing 4 MHz ultrasonic transducer with a near field length of $N = 67$ mm in water

Sound field of a non-focusing 4 MHz ultrasonic transducer with a near field length of $N = 67$ mm in water. The plot shows the sound pressure at a logarithmic db-scale.

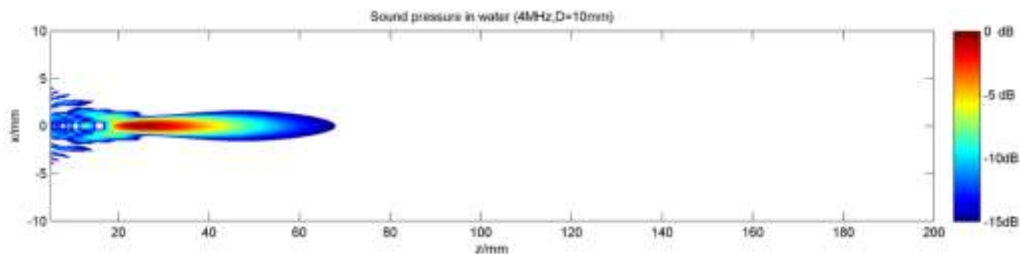


Figure (5.6): Sound pressure field of the same ultrasonic transducer (4 MHz, $N = 67$ mm) with the transducer surface having a spherical curvature with the curvature radius $R = 30$ mm

Sound pressure field of the same ultrasonic transducer (4 MHz, $N = 67$ mm) with the transducer surface having a spherical curvature with the curvature radius $R = 30$ mm

An ultrasonic transducer is a device that converts AC into ultrasound, as well as the reverse, sound into AC. In ultrasonics, the term typically refers to piezoelectric transducers or capacitive transducers. Piezoelectric crystals change size and shape when a voltage is applied; AC voltage makes them oscillate at the same frequency and produce ultrasonic sound. Capacitive transducers use electrostatic fields between a conductive diaphragm and a backing plate.

The beam pattern of a transducer can be determined by the active transducer area and shape, the ultrasound wavelength, and the sound velocity of the propagation medium. The diagrams show the sound fields of an unfocused and a focusing ultrasonic transducer in water, plainly at differing energy levels.

Chapter (5): Sensors

Since piezoelectric materials generate a voltage when force is applied to them, they can also work as ultrasonic detectors. Some systems use separate transmitters and receivers, while others combine both functions into a single piezoelectric transceiver.

Ultrasound transmitters can also use non-piezoelectric principles. Such as magnetostriction. Materials with this property change size slightly when exposed to a magnetic field, and make practical transducers.

A capacitor ("condenser") microphone has a thin diaphragm that responds to ultrasound waves. Changes in the electric field between the diaphragm and a closely spaced backing plate convert sound signals to electric currents, which can be amplified.

The diaphragm (or membrane) principle is also used in the relatively new micro-machined ultrasonic transducers (MUTs). These devices are fabricated using silicon micro-machining technology (MEMS technology), which is particularly useful for the fabrication of transducer arrays. The vibration of the diaphragm may be measured or induced electronically using the capacitance between the diaphragm and a closely spaced backing plate (CMUT), or by adding a thin layer of piezo-electric material on diaphragm (PMUT). Alternatively, recent research showed that the vibration of the diaphragm may be measured by a tiny optical ring resonator integrated inside the diaphragm (OMUS).

5.4 Smoke Detector



Figure (5.7): Smoke Detector Sensor

A smoke detector [\[18\]](#) is a device that senses smoke, typically as an indicator of fire. Commercial security devices issue a signal to a fire alarm control panel as part of a fire alarm system, while household smoke detectors, also known as smoke alarms, generally issue a local audible or visual alarm from the detector itself.

Smoke detectors are housed in plastic enclosures, typically shaped like a disk about 150 millimeters (6 in) in diameter and 25 millimeters (1 in) thick, but shape and size vary. Smoke can be detected either optically (photoelectric) or by physical process (ionization), detectors may use either, or both, methods. Sensitive alarms can be used to detect, and thus deter, smoking in areas where it is banned. Smoke detectors in large commercial, industrial, and residential buildings are usually powered by a central fire alarm system, which is powered by the building power with a battery backup. Domestic smoke detectors range from individual battery-powered units, to several interlinked mains-powered units with battery backup; if any unit detects smoke, all trigger even in the absence of electricity.

The US National Fire Protection Association estimates that nearly two-thirds of deaths from home fires occur in properties without working smoke detectors.

Chapter (6): Modules

6.1 Relay Module:



Figure (6.1): Relay Module

Relays work on electromagnetism, When the Relay coil is energized it acts like a magnet and changes the position of a switch. The circuit which powers the coil is completely isolated from the part which switches ON/OFF, this provides electrical isolation. This is the reason we can control a relay using 5V's from an arduino and the other end of it could be running a 230V appliance, the 230V end is completely isolated from the 5V arduino circuitry.

To connect the 4 Relay board [\[19\]](#) to an Arduino is very easy and allows you to turn on and off a wide range of devices, both AC and DC. The first to connections are the ground and power pins, you need to connect the Arduino +5v to the 4 Relay board VCC pin and the Arduino ground to the 4 Relay board GND pin. Then it's an only a matter of just connecting the communication pins, labeled IN1, IN2, IN3 and IN4, two 4 data pinson the Arduino. In the example code below we used Arduino pins 7, 8, 9, 10. It's a good idea to avoid using Data pins 0 and 1 as they are used by the Arduino for serial communication and can cause problems when uploading code to the Arduino.

The default state of the relay when the power is off for COMM (power) to be connected to NC (normally closed), this is the equivalent of setting the 4 Relay boards IN pin to HIGH (has +5v sent to it) It is a safety feature to not use the NC connector in-case you Arduino loses power it will automatically turns off all the devices connected to the relay. When you have something connected to the relays NO (Normally Open) connector and you set the corresponding IN pin to LOW (0v), power will flow in from the COMM connector and out of the NO connector powering your device.

6.2 Xbee:

Xbee [\[20\]](#) is the brand name of a family of form factor compatible radio modules from Digi International. The first XBee radios were introduced under the MaxStream brand in 2005 and were based on the IEEE 802.15.4-2003 standard designed for point-to-point and star communications at over-the-air baud rates of 250 kbit/s.

Two models were initially introduced — a lower cost 1 mW XBee and the higher power 100 mW XBee-PRO. Since the initial introduction, a number of new XBee radios have been introduced and all XBees are now marketed and sold under the Digi brand.

The XBee radios can all be used with the minimum number of connections — power (3.3 V), ground, data in and data out (UART), with other recommended lines being reset and Sleep. Additionally, most XBee families have some other flow control, input/output (I/O), analog-to-digital converter (A/D) and indicator lines built in. A version called the programmable XBee has an additional on-board processor for user's code. The programmable XBee and a surface-mount version of the XBee radios were both introduced in 2010.



Figure (6.2): Xbee module

Chapter (7): Protocols

7.1 I²C

I²C (Inter-Integrated Circuit) [\[21\]](#), pronounced I-squared-C, is a multi-master, multi-slave, single-ended, serial computer bus invented by Philips Semiconductor (now NXP Semiconductors). It is typically used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication. Alternatively, I²C is spelled I2C (pronounced I-Two-C) or IIC (pronounced I-I-C).

Since October 10, 2006, no licensing fees are required to implement the I²C protocol. However, fees are still required to obtain I²C slave addresses allocated by NXP.

Several competitors, such as Siemens AG (later Infineon Technologies AG, now Intel mobile communications), NEC, Texas Instruments, STMicroelectronics (formerly SGS-Thomson), Motorola (later Freescale, now merged with NXP), Nordic Semiconductor and Intersil, have introduced compatible I²C products to the market since the mid-1990s.

SMBus, defined by Intel in 1995, is a subset of I²C, defining a stricter usage of which. One purpose of SMBus is to promote robustness and interoperability. Accordingly, modern I²C systems incorporate some policies and rules from SMBus, sometimes supporting both I²C and SMBus, requiring only minimal reconfiguration either by commanding or output pin use.

7.1.1 Design

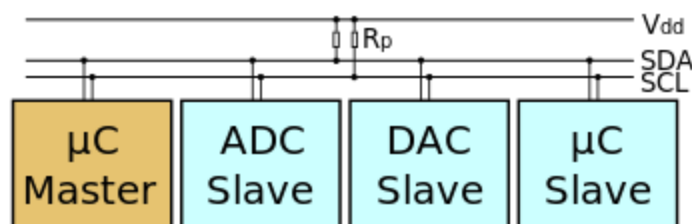


Figure (7.1): i2c design

A sample schematic with one master (a microcontroller), three slave nodes (an ADC, a DAC, and a microcontroller), and pull-up resistors R_p

I²C uses only two bidirectional open-drain lines, Serial Data Line (SDA) and Serial Clock Line (SCL), pulled up with resistors. Typical voltages used are +5 V or +3.3 V although systems with other voltages are permitted.

The I²C reference design has a 7-bit or a 10-bit (depending on the device used) address space. Common I²C bus speeds are the 100 kbit/s standard mode and the 10 kbit/s low-speed mode, but arbitrarily low clock frequencies are also allowed. Recent revisions of I²C can host more nodes and run at faster speeds (400 kbit/s Fast mode, 1 Mbit/s Fast mode plus or Fm+, and 3.4 Mbit/s High Speed mode). These speeds are more widely used on embedded systems than on PCs. There are also other features, such as 16-bit addressing.

Note the bit rates are quoted for the transactions between master and slave without clock stretching or other hardware overhead. Protocol overheads include a slave address and perhaps a register address within the slave device as well as per-byte ACK/NACK bits. Thus the actual transfer rate of user data is lower than those peak bit rates alone would imply. For example, if each interaction with a slave inefficiently allows only 1 byte of data to be transferred, the data rate will be less than half the peak bit rate.

The maximum number of nodes is limited by the address space, and also by the total bus capacitance of 400 pF, which restricts practical communication distances to a few meters. The relatively high impedance and low noise immunity requires a common ground potential, which again restricts practical use to communication within the same PC board or small system of boards.

7.1.2 Message protocols

I²C defines basic types of messages, each of which begins with a START and ends with a STOP:

Single message where a master writes data to a slave;

Single message where a master reads data from a slave;

Combined messages, where a master issues at least two reads and/or writes to one or more slaves.

In a combined message, each read or write begins with a START and the slave address. After the first START in a combined message these are also called repeated START bits. Repeated START bits are not preceded by STOP bits, which is how slaves know the next transfer is part of the same message.

Any given slave will only respond to certain messages, as specified in its product documentation.

Pure I²C systems support arbitrary message structures. SMBus is restricted to nine of those structures, such as read word N and write word N, involving a single slave. PMBus extends SMBus with a Group protocol, allowing multiple such SMBus transactions to be sent in one combined message. The terminating STOP indicates when those grouped actions should take effect. For example, one PMBus operation might reconfigure three power supplies (using three different I²C slave addresses), and their new configurations would take effect at the same time: when they receive that STOP.

With only a few exceptions, neither I²C nor SMBus define message semantics, such as the meaning of data bytes in messages. Message semantics are otherwise product-specific. Those exceptions include messages addressed to the I²C general call address (0x00) or to the SMBus Alert Response Address; and messages involved in the SMBus Address Resolution Protocol (ARP) for dynamic address allocation and management.

In practice, most slaves adopt request/response control models, where one or more bytes following a write command are treated as a command or address. Those bytes determine how subsequent written bytes are treated and/or how the slave responds on subsequent reads. Most SMBus operations involve single byte commands.

7.1.3 Circuit interconnections

I²C is popular for interfacing peripheral circuits to prototyping systems, such as the Arduino and Raspberry Pi. I²C does not employ a standardized connector, however, and board designers have created various wiring schemes for I²C interconnections. To minimize the possible damage due to plugging 0.1-inch headers in backwards, some developers have suggested using alternating signal and power connections of the following wiring schemes: (GND, SCL, VCC, SDA) or (VCC, SDA, GND, SCL).

Chapter (8): Implementation

SERIAL CONNECTION:

To send data from devices you need a connection to make this so we choose the Serial Connection.

This code makes a serial connection between Arduino, Raspberry Pi, and the Mobile Application to send and receive data from and to each device.

Android "Java":

```
if (bt_conn == false) {
    Set<BluetoothDevice> devices = bluetoothAdapter.getBondedDevices();
    if (devices != null) {
        for (BluetoothDevice device : devices) {
            if (deviceAddress.equals(device.getAddress())) {
                result = device;
                boolean connected=true;
                try {
                    socket =
result.createRfcommSocketToServiceRecord(PORT_UUID);
                    socket.connect();
                } catch (IOException e) {
                    e.printStackTrace();
                    connected=false;
                }
                if(connected)
                {
                    conn_ttl.setVisibility(View.VISIBLE);
                    srch_ttl.setVisibility(View.INVISIBLE);
                    splash_btn.setEnabled(true);

                    splash_btn.setBackgroundResource(R.drawable.btn_splash_nor);
                    try {
                        outputStream=socket.getOutputStream();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    try {
                        inputStream=socket.getInputStream();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    bt_conn = true;
                }
                break;
            }
        }
    }
}
```


Chapter(8): Implementation

Arduino C:

```
#include <SoftwareSerial.h>
SoftwareSerial BT(6, 7); //TX, RX
Serial.begin(9600);
BT.begin(9600);
ByteReceived = Serial.read();
```

Python:

```
import serial
ser = serial.Serial('/dev/ttyACM1', 9600)
while True:
    s[0] = str(int(ser.readline(),16))
```

CONTROL MODES “Manual - Auto”:

To give the freedom to the users, we made a two systems to control “Automatic Mode – Manual Mode”.

This code checks whether the mobile application is on to turn the system from automatic mode to manual mode and vice versa.

Android “Java”:

```
public static void appopn(){
    if (socket!=null)
    {
        try
        {
            socket.getOutputStream().write("9".toString().getBytes());
        }
        catch (IOException e){}
    }
}

public static void appcls(){
    if (socket!=null)
    {
        try
        {
            socket.getOutputStream().write("0".toString().getBytes());
        }
        catch (IOException e){}
    }
}
```

Arduino C:

```
if (char(ByteReceived) == '9' ) {
    mode=1;
} else if (char(ByteReceived) == '0' ) {
    mode=0;
}
```

Python:

```
def exit_handler():
    write_serial=ser.write('0')

atexit.register(exit_handler)

write_serial=ser.write('9')
```

SEND & RECEIVE BYTES:

This code convert the data from decimal to binary to send it or convert the data from binary to decimal to receive it.

Android "Java":

```
while (true) {
    if (bt_conn == true) {
        //read data from serial then convert it from decimal to char
        try {
            serial_msg = Integer.parseInt(new
String(String.valueOf(inputStream.read())));
        } catch (IOException e) {
            e.printStackTrace();
        }
        //send converted data to another string in another activity
        ActivityControlCenter.read_msg = String.valueOf((char)
serial_msg);
    }
}

public static void room1ON() {
    if (socket!=null)
    {
        try
        {
            socket.getOutputStream().write("5".toString().getBytes());
        }
        catch (IOException e){}
    }
}
```

Arduino C:

```
//office
if(char(ByteReceived) == '5' ) {
    digitalWrite(13, HIGH);
}
//gas sensor
if (gas == 1) {
    sprintf(dataString,"%02X",a);
    Serial.println(dataString);
    digitalWrite(redLed, HIGH);
    Serial.println("S");
    delay(3000);
}
```

Python:

```
s = [0]
while True:
    s[0] = str(int(ser.readline(),16))
    if (s[0] == "20"):
        ...
    time.sleep(20)
    s[0] = "30"
```

ROOMS STATUS:

To let the mobile application know the rooms status, so we need a method to check the rooms status.

This code checks whether the room is on or off, then send the room status to the mobile application.

Android "Java":

```
public void chk_on_off(){
    if (read_msg.matches("V")){
        room1_str = 1;
    }
    else if (read_msg.matches("v")){
        room1_str = 0;
    }
    if (read_msg.matches("K")){
        room2_str = 1;
    }
    else if (read_msg.matches("k")){
        room2_str = 0;
    }
    if (read_msg.matches("D")){
        room3_str = 1;
    }
    else if (read_msg.matches("d")){
        room3_str = 0;
    }
    if (read_msg.matches("F")){
        room4_str = 1;
    }
    else if (read_msg.matches("f")){
        room4_str = 0;
    }
    if (read_msg.matches("G")){
        room5_str = 1;
    }
    else if (read_msg.matches("g")){
        room5_str = 0;
    }
    if (read_msg.matches("H")){
        room6_str = 1;
    }
    else if (read_msg.matches("h")){
        room6_str = 0;
    }
}
```

Chapter(8): Implementation

Arduino C:

```
if (on_off_mode < 50) {  
    if (digitalRead(11) == HIGH) {  
        Serial.println("D");  
    }  
    else {  
        Serial.println("d");  
    }  
    if (digitalRead(12) == HIGH) {  
        Serial.println("G");  
    }  
    else {  
        Serial.println("g");  
    }  
    if (digitalRead(13) == HIGH) {  
        Serial.println("V");  
    }  
    else {  
        Serial.println("v");  
    }  
    if (digitalRead(8) == HIGH) {  
        Serial.println("H");  
    }  
    else {  
        Serial.println("h");  
    }  
    if (digitalRead(9) == HIGH) {  
        Serial.println("K");  
    }  
    else {  
        Serial.println("k");  
    }  
    if (digitalRead(10) == HIGH) {  
        Serial.println("F");  
    }  
    else {  
        Serial.println("f");  
    }  
}
```

GAS SENSOR:

One of our security systems is the gas and smoke detection.

This code checks whether the gas sensor digital pin is high or low.

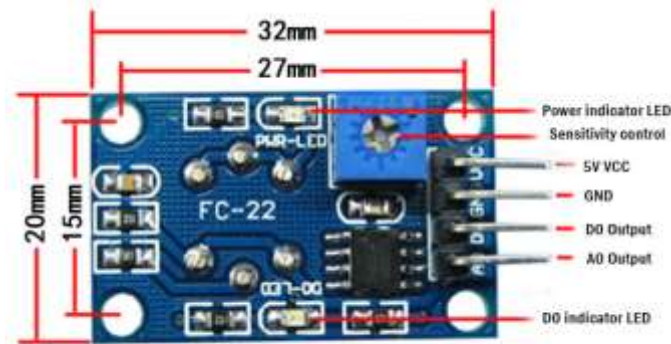


Figure (8.1): Gas Sensor Pinout

Android "Java":

```
public void smk_stt_th(){
    if (read_msg.matches("s")){
        smk_stt = false;
    }
    else if (read_msg.matches("S")){
        smk_stt = true;
    }
    final RelativeLayout ket_sec = (RelativeLayout)
this.findViewById(R.id.ket_smoke_sec);
    final TextView smk_sttText = (TextView)
this.findViewById(R.id.ket_smoke_st);
    if (smk_stt == true){
        smk_sttText.setText("YES");
        ket_sec.setBackgroundResource(R.drawable.btn_grid_erg);
    }
    else if (smk_stt == false){
        smk_sttText.setText("NO");
        ket_sec.setBackgroundResource(R.drawable.btn_grid_off);
    }
}
```

Chapter(8): Implementation

Arduino C:

```
if (digitalRead(gasPin) == LOW) {
    gas = gas + 1;
    delay(2000);
}
else{
    gas = 0;
    Serial.println("s");
}

if (gas == 1) {
    sprintf(dataString,"%02X",a);
    Serial.println(dataString);
    digitalWrite(redLed, HIGH);
    Serial.println("S");
    delay(3000);
}
```

Python:

```
import serial
import time

ser = serial.Serial('/dev/ttyACM1', 9600)

s = [0]

while True:

    s[0] = str(int(ser.readline(),16))

    if (s[0] == "20"):

        os.system('echo "This is warning" | mail-s "Smoke detected in your Office" EMAIL)

        time.sleep(20)
```

MOTION SENSOR:

One of our security systems is the motion detection.

This code checks the analog input from motion sensor analog pin.



- VCC-** Connects to 5V of positive voltage for power
- Trig-** A pulse is sent here for the sensor to go into ranging mode for object detection
- Echo-** The echo sends a signal back if an object has been detected or not. If a signal is returned, an object has been detected. If not, no object has been detected.
- GND-** Completes electrical pathway of the power.

Figure (8.2): Ultra-Sonic Sensor Pinout

Android "Java":

```
public void mo_stt_th(){
    if (read_msg.matches("w")){
        mo_stt = false;
    }
    else if (read_msg.matches("W")){
        mo_stt = true;
    }
    final RelativeLayout mo_sec = (RelativeLayout)
this.findViewById(R.id.room1_mo_sec);
    final TextView mo_sttText = (TextView)
this.findViewById(R.id.room1_mo_st);
    if (mo_stt == true){
        mo_sttText.setText("YES");
        mo_sec.setBackgroundResource(R.drawable.btn_grid_nor);
    }
    else if (mo_stt == false){
        mo_sttText.setText("NO");
        mo_sec.setBackgroundResource(R.drawable.btn_grid_off);
    }
}
```


Chapter(8): Implementation

Arduino C:

```
long duration, cm;

pinMode(trigPin, OUTPUT);
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

pinMode(echoPin, INPUT);
duration = pulseIn(echoPin, HIGH);

cm = microsecondsToCentimeters(duration);
if (cm < 20) {
    digitalWrite(light_1, HIGH);
    Serial.println("W");
}
else{
    digitalWrite(light_1, LOW);
    Serial.println("w");
}
```

TEMPERATURE SENSOR:

This heat sensor helped us to know the temperature of the control center and turn on a cooling fan if needed.

This code checks the analog input from temperature sensor analog pin.

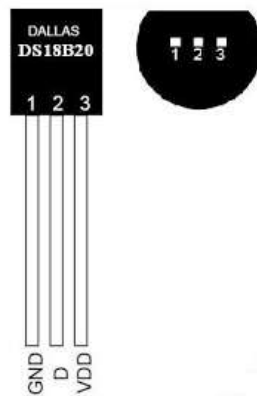


Figure (8.3): Temp sensor pinout

Android“Java”:

```
public void deg_text(){
    cc_degText = read_msg;
    final TextView deg_cc = (TextView) this.findViewById(R.id.cc_temp_st);
    if (cc_degText != null & cc_degText.length() > 0 &
    cc_degText.matches("[0-9]*$")) {
        deg_cc.setText(cc_degText + "0" + "°");
    }
    final RelativeLayout heat_stt_sec = (RelativeLayout)
    this.findViewById(R.id.heat_sec);
    if (cc_degText.matches("2")) {
        heat_stt_sec.setBackgroundResource(R.drawable.btn_grid_cold);
    }
    else if (cc_degText.matches("9")) {
        heat_stt_sec.setBackgroundResource(R.drawable.btn_grid_erg);
    }
    else {
        heat_stt_sec.setBackgroundResource(R.drawable.btn_grid_cold);
    }
}
```

Chapter(8): Implementation

Arduino C:

```
if (char(ByteReceived) == 'h') {
    digitalWrite(fanPin, HIGH);
    Serial.println("9");
}
else if (char(ByteReceived) == 'c') {
    digitalWrite(fanPin, LOW);
    Serial.println("2");
}
```

Python:

```
import time
import serial
ser = serial.Serial('/dev/ttyACM0',9600)
while 1:
    tempfile = open("/sys/bus/w1/devices/28-000002f53b3e/w1_slave")
    thetext = tempfile.read()
    tempfile.close()
    tempdata = thetext.split("\n")[1].split(" ")[9]
    temprature = float(tempdata[2:])
    temprature = temprature / 1000
    temprature = int(temprature)
    print temprature
    if (temprature > 29):
        write_serial=ser.write('h')
        print("fan on")
    else:
        write_serial=ser.write('c')
        print("fan off")
    time.sleep(1)
```

References

1. Embedded System:

https://www.tutorialspoint.com/embedded_systems/es_overview.htm

2. IOT:

<http://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#5ac78c8e6828>

| | |
|-----------|---|
| Title | From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence |
| Authors | Jan Holler , Vlasios Tsiatsis , Catherine Mulligan , Stefan Avesand , Stamatis Karnouskos , David Boyle |
| Publisher | Academic Press, 2014 |
| ISBN | 0080994016, 9780080994017 |
| Length | 352 pages |

3. Microcontroller:

<http://internetofthingsagenda.techtarget.com/definition/microcontroller>

4. Raspberry Pi:

https://en.wikipedia.org/wiki/Raspberry_Pi

| | |
|-----------|-------------------------------------|
| Title | Getting Started with Raspberry Pi 3 |
| Author | Agus Kurniawan |
| Publisher | PE Press, 2016 |

5. Arduino:

Title Beginning Android ADK with Arduino
 [Apress Series](#)
 [Technology in action series](#)

Author [Mario Böhmer](#)

Edition illustrated

Publisher Apress, 2012

ISBN 1430241977, 9781430241973

Length 316 pages

Basic Electronics:

Title Handbook of Modern Sensors: Physics, Designs, and Applications

Author [Jacob Fraden](#)

Edition 4, illustrated, reprint

Publisher Springer Science & Business Media, 2010

ISBN 1441964665, 9781441964663

Length 663 pages

6. <https://learn.sparkfun.com/tutorials/diodes>

7. <http://whatis.techtarget.com/definition/light-emitting-diode-LED>

8. <http://whatis.techtarget.com/definition/resistor>

9. <https://learn.sparkfun.com/tutorials/voltage-current-resistance-and-ohms-law>

10. <http://www.rapidtables.com/electric/capacitor.htm>

11. <http://whatis.techtarget.com/definition/transistor>

12. https://www.thebuilderssupply.com/Push-buttons-and-much-more_b_14.html

13. <http://electronics.howstuffworks.com/relay.htm/printable>

14. <https://www.ecse.rpi.edu/courses/F15/ENGR-2300/breadboard.pdf>

Sensors

| | |
|-----------|--|
| Title | Handbook of Modern Sensors: Physics, Designs, and Applications |
| Author | Jacob Fraden |
| Edition | 4, illustrated, reprint |
| Publisher | Springer Science & Business Media, 2010 |
| ISBN | 1441964665, 9781441964663 |
| Length | 663 pages |

15. -LDR

<http://www.electrical4u.com/light-dependent-resistor-ldr-working-principle-of-ldr/>

16. -PIR

<http://www.instructables.com/id/PIR-Motion-Sensor-Tutorial/>

17. -Ultrasonic Transducer

<http://www.ab.com/en/epub/catalogs/12772/6543185/12041221/12041229/print.html>

18. -Smoke Detector

<https://www.hackster.io/Aritro/smoke-detection-using-mq-2-gas-sensor-79c54a>

Modules

19. -Relay Module

<http://www.hobbyist.co.nz/?q=interfacing-relay-modules-to-arduino>

20. -Xbee

<https://en.wikipedia.org/wiki/XBee>

Protocols

21. -I²C

<http://www.i2c-bus.org/>