UNIT II: Elements of IoT

Hardware and software devices combine to form an IoT ecosystem. Hardware is a set of devices that wire together to serve some functionality. Assume to build a drone, attach sensors to this drone so that it can take photos of your agricultural crops to keep a track of their growth. Or a smart watch that keeps a track of your entire schedule, count the numberof steps you take daily, measure your heart pulse. These examples will require connecting small components, tracking the battery usage. IoT devices are a combination of hardware and software. The two components integrated and perform a variety of functions.

Building blocks of IoT Hardware

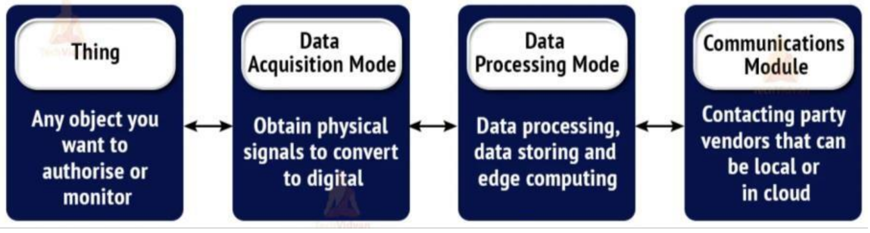

Thing
Any object you want to authorise or monitor
Data Acquisition Mode
Obtain physical signals to convert to digital
Data Processing Mode
Data processing, data storing and edge computing
Communications Module
Contacting party vendors that can be local or in cloud

Figure 1: Building blocks of IoT Hardware

1. "**Things**": Things in IoT are any devices that are capable of connecting to the internet. They can transmit, retrieve and store data that they collect from the surrounding. They include home appliances such as geysers, microwaves, thermostats and refrigerators, etc.
2. **Data Acquisition module:** As the term suggests, this module is responsible for acquiring data from the physical surroundings or environment (changes in the temperature, movement, humidity and pressure) and convert the signal to digital from.
3. **Data processing module**: This module includes computers that process the data acquired from the previous module. They analyse the data, store data for future references and other purposes.
4. **Communication module**: This is the final building block and this module is responsible for communication with third party vendors. This could include device to device, device to server or device to user.

Types of IoT hardware

It is easier to develop an IoT application these days due to the ease in the availability of boards, Integrated circuits, prototype kits and platforms. These hardware components are low cost and reliable. they offer flexibility and the choice to design custom sensors with specific applications. At the same time you can also specify the networking area, data management and other functionalities as you want. There is a wide range of hardware components to choose from and based on your requirements you can pick the one that matches your proptype perfectly.

a. Microcontrollers

i. Microcontrollers are a type of SoC that provides data processing and storage units. They contain a processor for processing, ROM for storage. When you start building an IoT system, you must pick a microcontroller that fits your desired purpose. You might have to look at its datasheet to understand the properties and specifications.

ii. Microcontrollers have properties such as Datapath Bandwidth. Datapath Bandwidth specifies the number of bits in the registers. More bits, more accurate results.

iii. Microcontrollers connect to all the components of the system and thus they must have sufficient input/output pins. Microcontrollers must have performance depending on whatsystem you are developing.

iv. Microcontrollers use a communication protocol to communicate with one another. The protocols are helpful when you are building bigger systems that require constant communication with other devices.

b. Single-Board Computer(SBC)

i. Single Board Computers (SBC) that contain all the processing and computing properties of a computer on a single board. SBCs have memory units to store code, data, input, output unitsand microprocessors for computing. It also includes an in-built RAM.

ii. They are a preferred choice in IoT industrial applications as they improve the functionality of a regular computer, they are easily available and reduce the cost of transportation.

iii. Based on the kind of project you are making, you choose a SBC that fits into all your needs for that specific project. SBCs are ready made and available in the market at cheap prices as compared to desktops and computers.

iv. The types of SBCs commonly available in the market are Raspberry Pis, Arduino, Beagleboard and Qualcomm DragonBoard 410c.

IoT Software

A overview of IoT Softwares are-

1. C & C++: The C programming language has its roots in embedded systems—it even gotits start for programming telephone switches. It’s pretty universal, it can be used almost everywhere and many programmers already know it. C++ is the object-oriented version of C, which is a language popular for both the Linux OS and **Arduino** embedded IoT software systems. These languages were basically written for the hardware systems, which makes them so easy to use.

2. Java: While C and C++ are hardware specific, the code in JAVA is more portable. It is more like a write once and read anywhere you want.

3. Python: There has been a recent surge in the number of python users and has now become one of the “go-to” languages in Web development. It is slowly spreading to the embedded control and IoT world—specially the Raspberry Pi processor. Python is an interpreted language, which is, easy to read, quick to learn and quick to write. Also, it’s a powerhouse forserving data-heavy applications.

4. B#: Unlike most of the languages mentioned so far, B# was specifically designed for embedded systems having less memory size.

Hardwarecomponents

Embedded system denotes a system that embeds software into a computing platform. Thesystem is dedicated for either an application(s), specific part of an application, product or acomponentofalargesystem.

Embedded device refers to a device, which embeds software into a computing platform and performs the computations and communication for specific systems.

Microcontroller unit (MCU) means a single-chip VLSI unit (also called microcomputer), which may be having limited computational capabilities. The MCU possesses memory, flash, enhancedinput-outputcapabilitiesandanumberofon-chipfunctionalunits.

Flash memory is an electronic non-volatile computer memory storage medium that can beelectrically erased and reprogrammed. The two main types of flash memory, NOR flash andNAND flash, are named for the NOR and NAND logic gates. Both use the same cell design,consistingoffloating gateMOSFETs.

Timer refers to a device which enables initiating new action(s) on timer start, on the clock inputs, timeouts or when the number of clock inputs equal to ap reset value.

Port refers to a device that enables input output (IO) communication between the MCU andanotherdevicesuchasasensororactuatororkeypadorwithanexternalcomputingdevice.

USBport connectsthe device hardwareto acomputer,downloads thedeveloped codes intothe device from the computer,or sends the codes from the device to the computer. USB portcan also provide the power for charging the battery of the connected platform, thus an externalchargerisnotneeded.

GPIO pins refer to General Purpose Input-Output pins. A pin that can be used in addition todigital input and output for other purposes, such as Rx and Tx or SDA and SCK, PWMs, analoginputs, outputs or timer outputs. The Rx and Tx pins are used during UART protocol-basedreception and transmission,SDA and SCK areused during useof I2C protocol-based serialdata andclockcommunication.

RxReceiverTx

Transmitter

SDASerialDataSC

KSerialClock

I2CInter-IntegratedCircuit

UARTUniversalAsynchronousReceiverandTransmitterMC

UMicrocontrollerUnit

SoCSystemonachip

Board is an electronic hardware—an electronic circuit board with MCU or SoC, circuits andconnectors, which provide the connections to other ICs and circuit components. The ICs andcircuit components can also be inserted or joined or put in place onto the board, by surfacemounttechnology.Theboardmayalsohavebattery,powersupply,voltageregulatoror connectionsforthepower.

Platform denotes a set consisting of computing and communication hardware, software andoperating system (OS). A platform enables working with different software, APIs, IDEandmiddleware. A platform may enable the development of codes at the development stage. It mayalsoenableprototypedevelopmentforanapplication(s)orspecificpartsofanapplication.

Module (hardware) is smaller form-factor hardware which can be placed onto a board. Themodule may embed the software. It may enable use of the board circuit with shorter form factor.AnexampleisRFmoduleplaced ontoanelectronicboard.

Shieldmeansasupportingcircuitwithconnectionpins,socket(s)andsupportingsoftware.

The supporting circuit enables the connectivity of a board or computing platform to externalcircuits. The circuit connects the elements that can be plugged onto a board or platform. Usageof the supporting circuit provides extra features, such as connectivity to wireless devices, suchas ZigBee, ZigBee IP, and Bluetooth LE, Wi-Fi or GSM or RF module or to a wired device,such as Ethernet shield. The Ethernet shield enables wired connection of a platform to theEthernet controller and throughan external standard LAN socket enables connectivity to awired or Wi-Fi modem for the Internet. Shield is the term used in Arduino hardware for thesupportingcircuits.

Header means plastic-coated strip or plastic-capped plug-in which is placed on top of the pinholeswhenmakingconnectionofthewireswithoutelectronicsoldering.Aheaderalsoprovides jumpers.Six-pinheader meansplastic 6-pin plug-inthat connectsthe6 pin holes.This header is a component, distinct from enveloping words in a data stack for communicationatalayer accordingtoaprotocol.

Jumper denotes a wire with a solid tip at each end which is normally used to interconnect thecomponentsonanelectronic-circuitbreadboard.JumpersareusedfortransferringIOsorsignalstoorfromthepins.

Interrupt means an action in which a running program interrupts an hardware signal, such astimer timeout or on execution of a software instruction for interrupt. For example, a programinterrupts for sending the data or for accepting a newly added device in the system or onexecutionoftheINTinstruction.

IntegratedDevelopmentEnvironment(IDE)meansasetofsoftwarecomponentsandmoduleswhichprovidethesoftwareenvironmentfordevelopingandprototyping.

Operatingsystem(OS)isasystemsoftwarewhichfacilitatestherunningofprocesses,allocation of memory, system calls to the IOs, facilitates the use of network subsystems, andwhich does devices management, priority allocations of processes and threads, and Prototypingthe Embedded Devices for IoT and M2M 295 enables multitasking and running of number ofthreads.TheOSenablesmanysystemfunctionsusingthegivencomputingdevicehardware.

IoT Hardware Devices

1. **Sensors:** A sensor is an IoT device that senses physical changes in the environment and sends the data for manipulation via a network. Clouds store the data for future references. Sensors monitor data and collect information constantly.

2. **Microcontrollers:** A microcontroller is a small computer that is capable of performing operations. It sits on a semiconductor integrated circuit chip. Microcontollers usually operate on a single function and hence differ from regular computers. They perform a variety of tasksin a relatively simpler manner.

3. **Wearable devices:** Wearable devices are a benchmark revolution of the IoT industry. Theseare Iot devices that humans can wear on their bodies to regulate and perform a variety of tasks. These wearables are capable of tracking glucose levels, monitor heart attack risks, coagulation and asthma monitoring, daily step and calorie consumption tracking.

4. **Basic devices:** Traditional computers such as desktops, tablets and cellphones are still an integral part of any IoT ecosystem. Desktops offer users with simple access to a lot of information and cell phones allow remote access to Iot devices using APIs.

5. **Datasheets:** Datasheets give the details about the functionality of any hardware components. It is important to study the datasheet of any hardware before making a purchase to make sure you are buying the right product.

   Datasheets offer you detailed information on the parameters of the hardware, its physical size, different voltage and electrical parameters, maximum current usage and the number of input/output pins. Datasheets are highly useful as they give you all the information you need before buying complicated hardware components.

6. **Integrated circuits:** Integrated circuits are chips. They are microcontrollers. You can buy empty chips in the market and download any kind of design into the chip. They are made using Silicon and it is packaged into shapes of rectangles. These chips contain complicated logic circuits, gates, registers, switches, I/O terminals and flip flops.

   Integrated circuits do a variety of functions, they can perform arithmetic and logical calculations. They act as processors too. They contain binary coded information which isprogrammed to perform a set of tasks.

   Standard chips are available in the market that perform a fixed set of operations. You can alsoconstruct chips to perform your desired set of functions and these are known as custom made chips.

IoT Hardware Providers

Various companies have come up with their own personalized IoT hardware and softwareand many emerging companies are adapting to these policies. However, the most common Iothardware providers are listed below:

a. Adafruit is best if you want to get hands-on experience with IoT. The company sells IoT DIY kits with an online guide to help you through the initial setting up. You can interact, manipulate and store your data.

b. Raspberry Pi is best at student level to get hands-on experience with IoT. You can interact, manipulate and store your data.

b. Arduino, the company brands microcontrollers, IoT kits and software tools.

c. Lantronix provides solutions for the IoT such as smart hardware, networking, engineering and artificial intelligence.

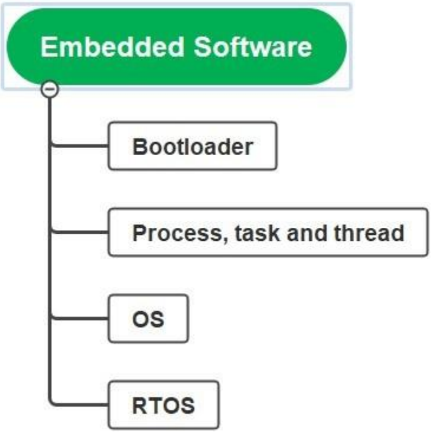
d. Espressif can interconnect with the system to provide wifi and bluetooth. It has high level integration. It uses low power and has a robust design.

COMPUTING

EMBEDDEDCOMPUTINGBASICS

Embedding means embedding function software into a computing hardware to enable a systemfunction for the specific dedicated applications. A device embeds software into the computingandcommunicationhardware,andthedevicefunctionsfortheapplications.

8.2.1 EmbeddedSoftware



Bootloader

Bootloader is a program which runs at the start of a computing device, such as an MCU. Abootloaderinitiatesloadingofsystemsoftware(OS)whenthesystempowerisswitchedon,and power-on-selftestcompletes.

OperatingSystem

An operating system (OS) facilitates the use of system hardware and networking capabilities.When a load oftheOS intoRAMcompletes then theMCU startsthenormaloperationalruntimeenvironment.Whenthedeviceisexecutingmultipletaskso rthreads,thenalsoanOSisrequired.TheOScontrolsthemultipleprocessesanddevicefunctions.

Process,taskandthreadarethesetofinstructionswhichrununderthecontroloftheOS.

The OS enables memory allocation to different processes, and prioritising of the processesenables the use of network hardware and device hardware functions and execution of softwarecomponentsandprocesses.

The OS is at flash memory of the device. It may be required to load functions at the deviceRAM.

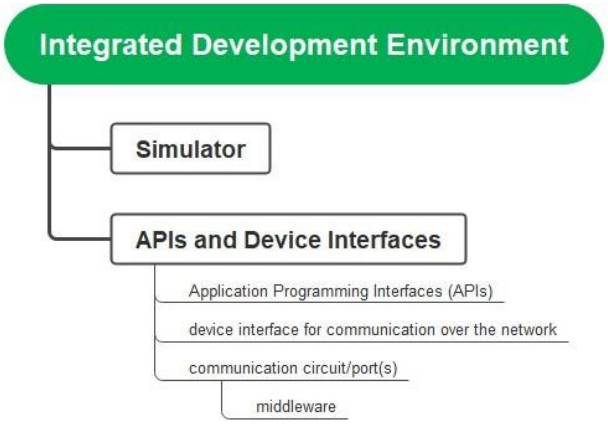
An OS may be open source, such as Linux or its distribution. Linux distribution means apackage or set of software components and modules bundled together for specific functions orforspecifichardware,anddistributedforwiderusagesandapplications.

Forexample,ArduinoLinuxdistributionrunsinArduinocircuitboardsandtheLinuxfunctionse nabledevelopingtheapplicationprogramsforusingtheArduino.

Real-TimeOperatingSystem

Real-TimeOperatingSystem(RTOS)isanOSthatenablesreal-timeexecutionofprocesseson computing and communication hardware. RTOS uses prioritisation and priority allocationconcepttoenabletheexecutionofprocesses inreal-time.

8.2.2 IntegratedDevelopmentEnvironment


Integrated Development Environment
Simulator
APIs and Device Interfaces
Application Programming Interfaces (APIs)
device interface for communication over the network
communication circuit/port(s)
middleware

IntegratedDevelopment Environment(IDE) is aset of software componentsandmoduleswhich provide the software and hardware environment for developing and prototyping. An IDEenables the codes development on a computer, and later on enables download of the codes onthe hardware platform.IDE enables software that communicates with theInternet web serverorcloudservice.

IDEconsists ofthedevice APIs,libraries,compilers,RTOS, simulator, editor, assembler,debugger,emulators,logicanalyser,and application codes' burner forflash,EPROMandEEPROMandothersoftwarecomponentsforintegrateddevelopmentofa system.IDEmaybe open source.For example,Arduino has open source IDE from the Arduinowebsite.TheIDEorprototypetoolenablesaprototypedesign.IDEisused fordeveloping embeddedhardwareandsoftwareplatforms,simulating,anddebugging.IDEisatoolforsoftwaredevelopment of embedded devices, which makes application development controllersystem.The library consists of a number of programs. The library has programs for each serial-interfaceprotocol, which can be used in the device. The program enables using the protocol-specificprograms directly, such as using a program for reading an RFID tag or using a program forsendingdatatotheUSBportforonwardtransmissionontheInternet.

Simulator

Simulatorissoftwarethatenablesdevelopmentonthecomputerwithoutanyhardware,andthenprototypinghardwarecanbeconnectedforembeddingthesoftwareandfurthertests.

APIsandDeviceInterfaces

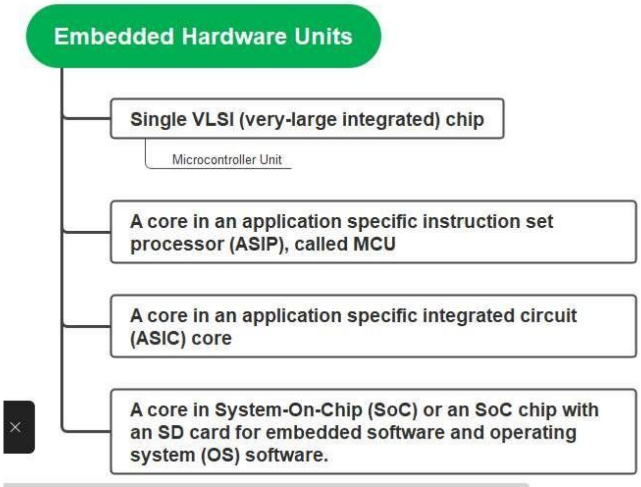
Softwareconsistsof device Application ProgrammingInterfaces (APIs)anddevice interfacefor communication over the network and communication circuit/port(s) which also includes amiddleware.

The middleware creates IPv4, IPv6, 6LowPAN, MQTT, COAP, LWM2M, REST and othercommunicationprotocol stacks.

DeviceInterfaces

A connectivity interfaceconsistsofcommunicationAPIs,deviceinterfacesand processingunits. Software commands the action on the message or information received followed byhardwareportoutputs fortheactuators.

8.2.3 EmbeddedHardwareUnits


Embedded Hardware Units
Single VLSI (very-large integrated) chip
Microcontroller Unit
A core in an application specific instruction set processor (ASIP), called MCU
A core in an application specific integrated circuit (ASIC) core
A core in System-On-Chip (SoC) or an SoC chip with an SD card for embedded software and operating system (OS) software.

Thehardwareincludesthefollowing:

- SingleVLSI(very-largeintegrated)chip
- Acoreinanapplicationspecificinstructionsetprocessor(ASIP),calledMCU
- Acoreinanapplicationspecificintegratedcircuit(ASIC)core
- AcoreinSystem-On-Chip(SoC)oranSoCchipwithanSDcardforembeddedsoftwareandoperatingsystem(OS)software.

FollowingsubsectionsdescribeMCU,SoCandselectionofplatformforprototyping.

MicrocontrollerUnit

An MCU is a single-chip VLSI unit (also called microcomputer) which though having limitedcomputational capabilities, possesses enhanced input-output capability and has a number ofon-chip functional units, such as Internal RAM, flash, IO ports, GPIOs, serial interfaces, timers,serialportsandtimers.

Application-specificMCUshaveadditionalon-chipfunctionalunits,suchasPWMcircuits(1,2 or 3), ADC (1, 2, 4 or higher) and other functional units. Figure 8.1 shows the on-chipfunctionalunitsinamicrocontroller.

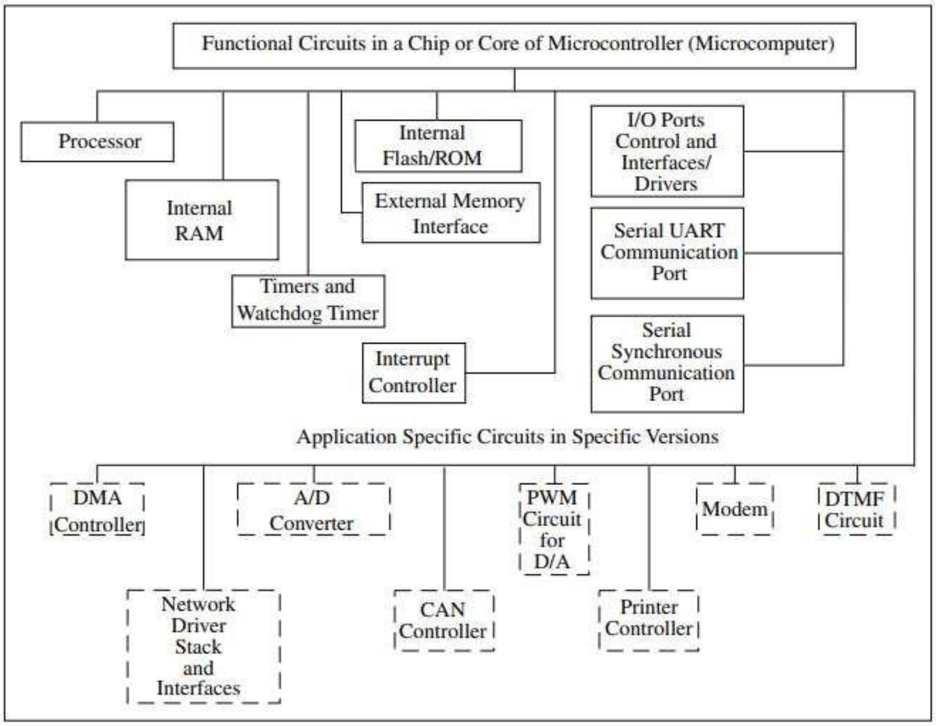

Functional Circuits in a Chip or Core of Microcontroller (Microcomputer)
Processor
Internal RAM
Timers and Watchdog Timer
Internal Flash/ROM
External Memory Interface
Interrupt Controller
I/O Ports Control and Interfaces/ Drivers
Serial UART Communication Port
Serial Synchronous Communication Port
Application Specific Circuits in Specific Versions
DMA Controller
A/D Converter
PWM Circuit for D/A
Modem
DTMF Circuit
Network Driver Stack and Interfaces
CAN Controller
Printer Controller

Figure 8.1 Microcontroller, on-chip functional units and application specific units

An MCU is an IC chip, available from a number of sources, such as ATMEL,Nexperia,Microchip, Texas Instruments or Intel. A source may manufacture different types, families andgroups of MCUs, like ATMEL manufactures AVR®8 and AVR®32 families of MCUs. AfamilyofMCUscanhavedifferentversions.

MCUcanbeof8-bit,16-bitor32-bitfamily.

● MCU clock frequency can be 8 MHz, 16 MHz, 100 MHz, 200 MHz or higher. The clockfrequency depends on the version and family. Performance defines number of instructionsexecutedpersecthatprimarilydependsontheclockfrequencyalso.Ametricforperformanceis Million Instructions Per Second (MIPS). Another metric for performance is Million FloatingPointOperationsPerSecond(MFLOPS).

● MCU includes RAM which can be 4 kB, 16 kB, 32 B or higher. RAM read and write of bytetakes an instruction cycle each. RAM is used for temporary variables, stacks and MCU includesEEPROMandflashmemory,whichmaybe512B,1kB,2kB,4kB,16kB,64kB,128kB,512 kB or higher. Flash stores the programs, data, tables and required information duringbuilding and testing stages, and then stores a final version of the application program in theembeddeddevice.

- MCU includes timers, I/O ports, GPIO pins, serial synchronous and asynchronous ports andinterruptcontrollers.
- MCU includes several functional units in specific version, such as ADC, multi-channel ADCor ADC with programmable positive and negative reference voltage pins, PWMs, RTC, I2C,CAN and USB ports, LCD interface, ZigBee interface, Ethernet, modem or other functionalunits,dependingonaspecificsource, family,groupandversion.

System-on-Chip

Complexembeddeddevices,suchasmobilephones,consistofacircuitwhichisdesignedona single silicon chip. The circuit consists of multiple processors, hardware units and software.An SoC is a system on a VLSI chip that has multiple processors, software and all the neededdigital as well as analog circuits' on chip. An SoC embeds processing circuit with memory andisspecifictodedicatedapplication(s)andmayhavetheanalogcircuits.

SoCmayassociateanexternalSDcardinamobilephone.Thecard storestheexternalprograms and operating system and enables the use of the chip distinctly for distinct purposes.SecureDigitalAssociationcreatedtheSDIO(SecureDataInput-Output)card.

The card consists of standard, mini, micro or nano form. It consists of flash memory andcommunication protocol. An SoC can be from different sources,for examples,Raspberry PiandBeagleBone.

SelectionofEmbeddedPlatform

Selection among the number of different availableplatforms depends on a number of factorslike price, open source availability, ease of application development and needed capabilities,performance required from IoT device and suitability for developing and using for prototypingand**designing.**

Hardware

Thechoice,besidetheprice,ofembeddedhardwaredependsonthefollowing:

- Processor speed required which depends upon the applications and services. For example,imageandvideoprocessingneedthehigh-speedprocessors
- RAM need which may be 4 kB or higher depending upon the OS and applications. Forexample,requirementis256kB forusingthe Linuxdistribution.NewgenerationmobilephoneshaveoveroneGBRAM.
- ConnectionneedstoZigBee,ZigBeeIP,BluetoothLE,Wi-FiorWiredEthernetfornetworking usingasupporting circuit (shield)
- USBhost
- Sensor, actuatorand controllers interfacing circuits,such as ADC,UART,I2C,SPI,CANsingleor multiples
- Powerrequirements,V-

andV+,0Vand3.3Vor0Vand5Vorother.SoftwarePlatformsandComponents

Selection among a number of different available software depends on hardware platform, opensourceavailabilityofsoftwarecomponents,costofavailabilityordevelopmentofotherrequired componentsforapplicationsandservices.

The choice of embedded software, beside the price, depends on the following: IDE with deviceAPIs,libraries,OSorRTOS,emulator,simulatorandotherenvironmentcomponents,middleware with communication and Internet protocols, and Cloud and sensor-cloud platformforapplicationsdevelopment,datastorageand services.

OpenSourceFrameworkforIoTImplementation

EclipseIoT(www.iot.eclipse.org)providesopensourceimplementationsofanumberofstandards including MQTT CoAP, LWM2M and services and frameworks that enable an OpenInternet of Things software. Eclipse tool work with Lua. An IoT programming language is Lua.IoT can beprogrammed inany open source languagelikePythonor Javaor PHPalso usingthetools.

Arduino development tools provide a set of software that includes an IDE and the Arduinoprogramming language for ahardware specification for interactive electronics that can senseandcontrol moreofthephysicalworld.

Middleware

OpenIoT is an open source middleware. It enables communication with sensor clouds andenables cloudbased ‘sensing as a service’, IoTSyS is middleware. It enables provisioning ofcommunicationstackforsmartdevicesusingIPv6andanumberofotherstandardprotocols.

WebServices

Web server or cloud server or clients use SOAP, REST, JSON, HTTP, HTTPS, web sockets,web APIs, URI. These provide the building blocks (software components) for writing the codesforWebServices.

CloudApplicationsDevelopmentPlatforms

The cloud-based development platforms are also widely used for IoT because of world wideavailability,storageandplatformspecificfeatures.

Xively (Pachube/Cosm) is another cloud application development platform for sensor data.XivelyisopensourceforbasicservicesItissoftwareandserverplatformfordatacapture,datavisualizationreal-timeandotherfeaturesovertheInternet.ItisanopensourceplatformforArduinoopensourceelectronicsprototypingplatformconnectivitywithweb.

NimbitsenablesIoTon an open source distributed cloud. PaaS atthe Nimbits cloud deploysaninstanceofNimbitsserveratthedevicenodes.

Arduino

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to asa microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board -- you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

How arduino works?

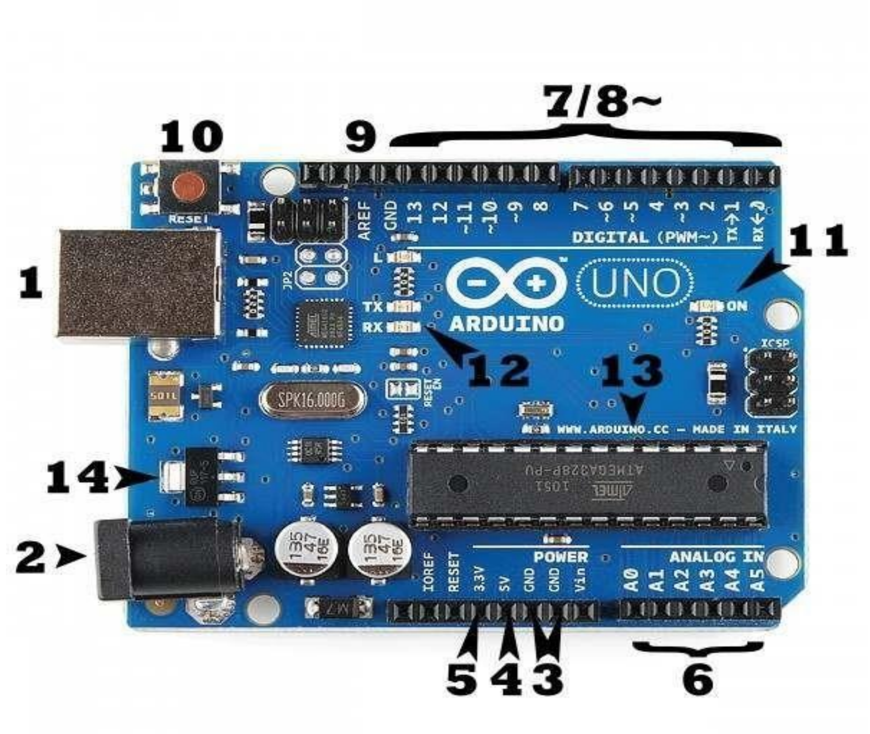
The Arduino hardware and software was designed for artists, designers, hobbyists, hackers, newbies, and anyone interested in creating interactive objects or environments. Arduino can interact with buttons, LEDs, motors, speakers, GPS units, cameras, the internet, and even your smart-phone or your TV! This flexibility combined with the fact that the Arduino software is free, the hardware boards are pretty cheap, and both the software and hardware are easy to learn has led to a large community of users who have contributed code and released instructions for a **huge** variety of Arduino-based projects.

For everything from robots and a heating pad hand warming blanket to honest fortune-telling machines, and even the Arduino can be used as the brains behind almost any electronics project.

Arduino board

There are many varieties of Arduino boards (explained on the next page) that can be used for different purposes. Some boards look a bit different from the one below, but most

Arduinos have the majority of these components in common:



- ***Power (USB / Barrel Jack):*** Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply (like this) that is terminated in a barrel jack. In the picture above the USB connection is labeled **(1)** and the barrel jack is labeled **(2)**.

- ***Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF):*** The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjuction with a breadboard and some wire. They usually have black plastic 'headers' that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

  - ✓ **GND (3)**: Short for 'Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
  - ✓ **5V (4) & 3.3V (5)**: As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
  - ✓ **Analog (6)**: The area of pins under the 'Analog In' label (A0 through A5 on the UNO) is Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
  - ✓ **Digital (7)**: Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).

- ✓ **PWM (8)**: You may have noticed the tilde (~) next to some of the digital pins (3, 5,6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (likefading an LED in and out).
- ✓ **AREF (9)**: Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

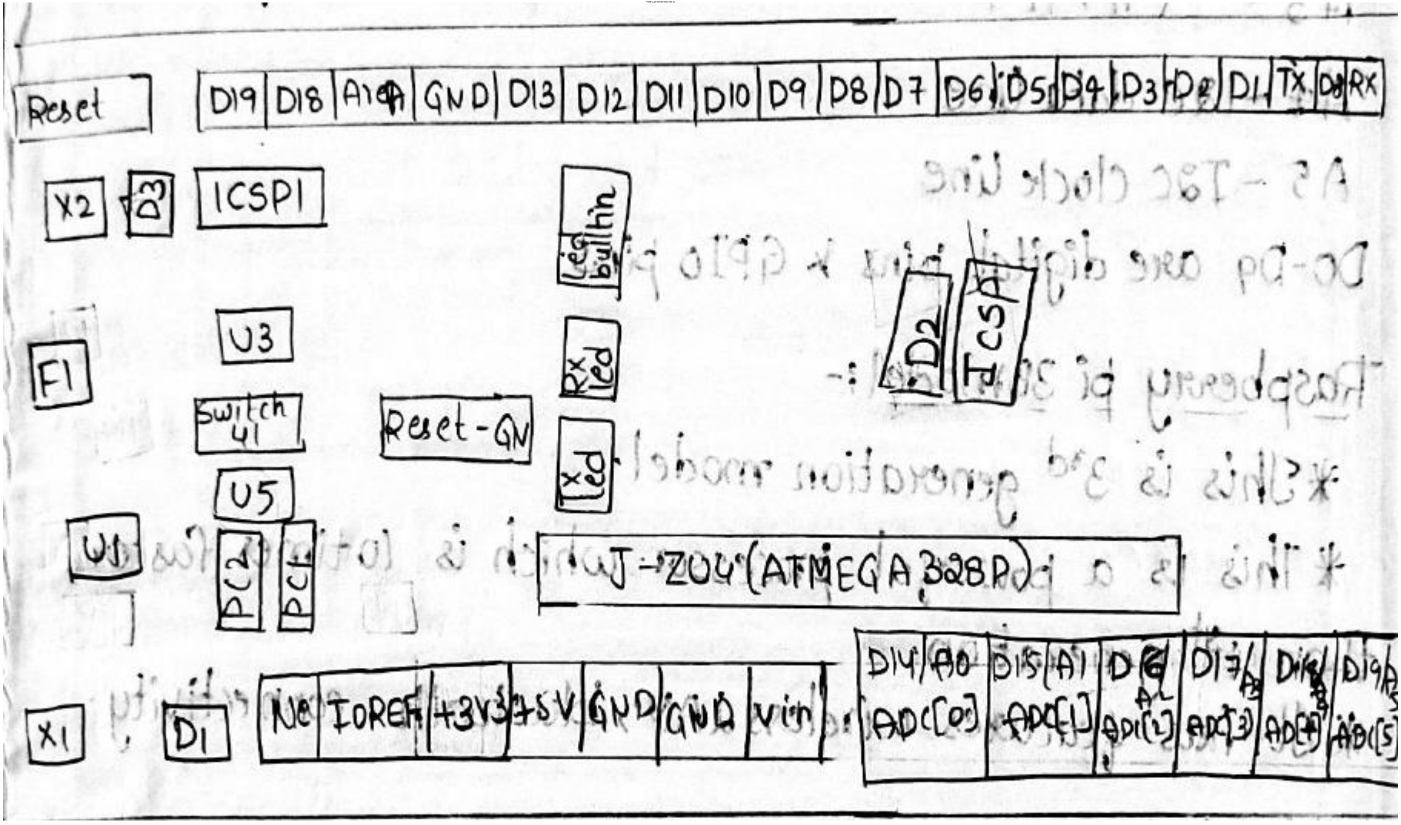
- ***Reset Button:*** Just like the original Nintendo, the Arduino has a reset button **(10)**. Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn't usually fix any problems.

- ***Power LED Indicator:*** Just beneath and to the right of the word "UNO" on your circuitboard, there's a tiny LED next to the word 'ON' **(11)**. This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong.

- ***TX RX LEDs:*** TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear -- once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs **(12)**. These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

- ***Main IC:*** The black thing with all the metal legs is an IC, or Integrated Circuit **(13)**. Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC's from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC's, reading the datasheets is often a good idea.

- ***Voltage Regulator:*** The voltage regulator **(14)** is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it's for. The voltage regulator does exactly what it says -- it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper;it will turn away an extra voltage that might harm the circuit. Of course, it has its limits,so don't hook up your Arduino to anything greater than 20 volts.



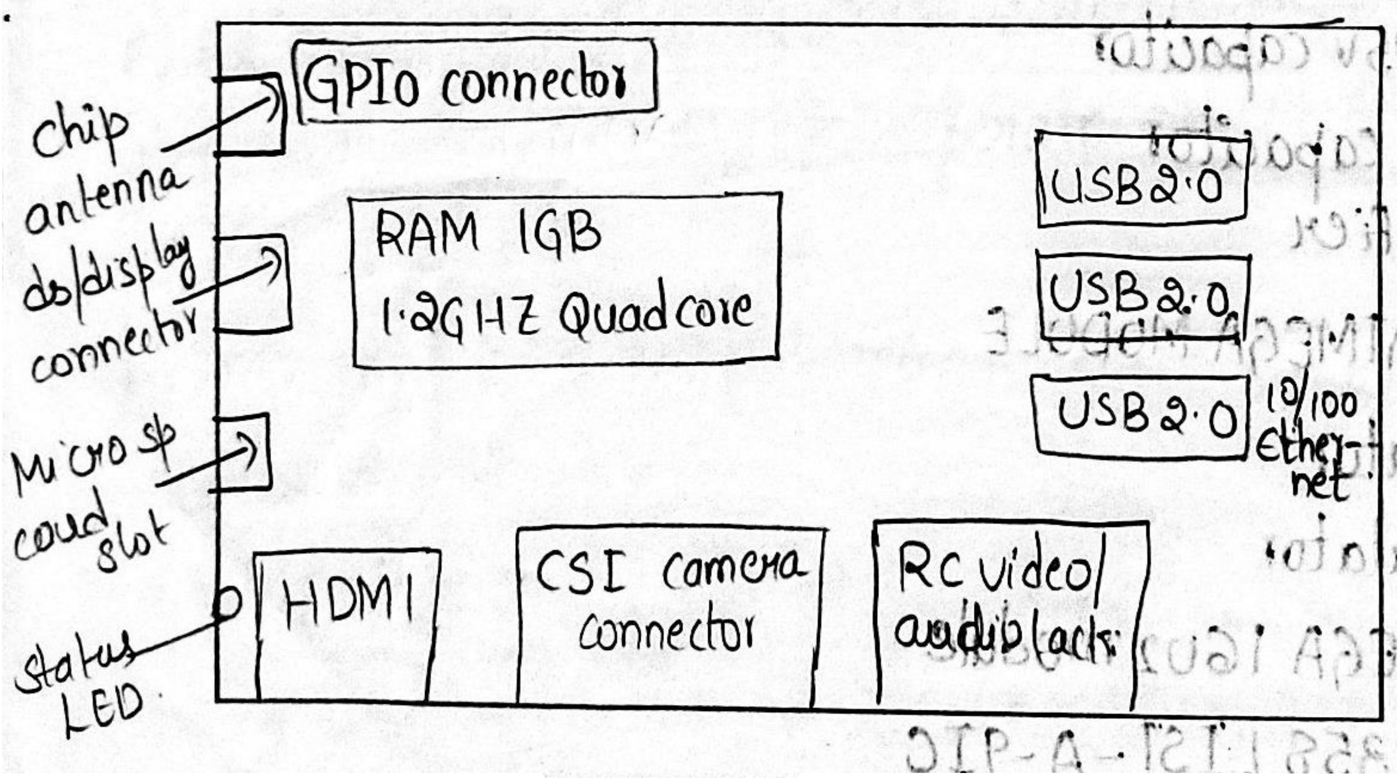
- Raspberry pi

The Raspberry pi is a single computer board with credit card size, that can be used for many tasks that your computer does, like games, word processing, spreadsheets and alsoto play HD video. It was established by the Raspberry pi foundation from the UK. It has been ready for public consumption since 2012 with the idea of making a low-cost educational microcomputer for students and children. The main purpose of designing the raspberry pi board is, to encourage learning, experimentation and innovation for school level students. The raspberry pi board is a portable and low cost. Maximum of the raspberry pi computers is used in mobile phones. In the 20th century, the growth of mobilecomputing technologies is very high, a huge segment of this being driven by the mobile industries. The 98% of the mobile phones were using ARM technology.

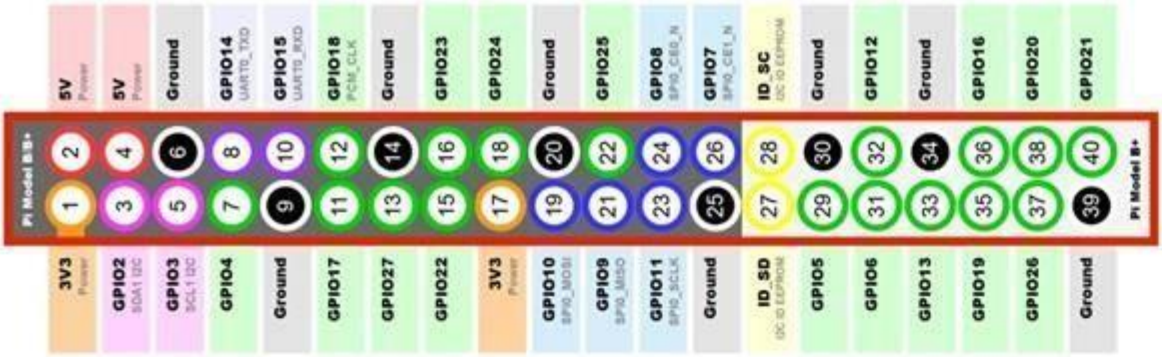
Raspberry Pi Hardware Specifications

The raspberry pi board comprises a program memory (RAM), processor and graphics chip, CPU, GPU, Ethernet port, GPIO pins, Xbee socket, UART, power source connector and various interfaces for other external devices. It also requires mass storage, for that we use an SD flash memory card. So that raspberry pi board will boot from this SD card similarly as a PC boots up into windows from its hard disk

Essential hardware specifications of raspberry pi board mainly include SD card containing Linux OS, US keyboard, monitor, power supply and video cable. Optional hardware specifications include USB mouse, powered USB hub, case, internet connection, the Model A or B: USB WiFi adaptor is used and internet connection to Model B is LAN cable.

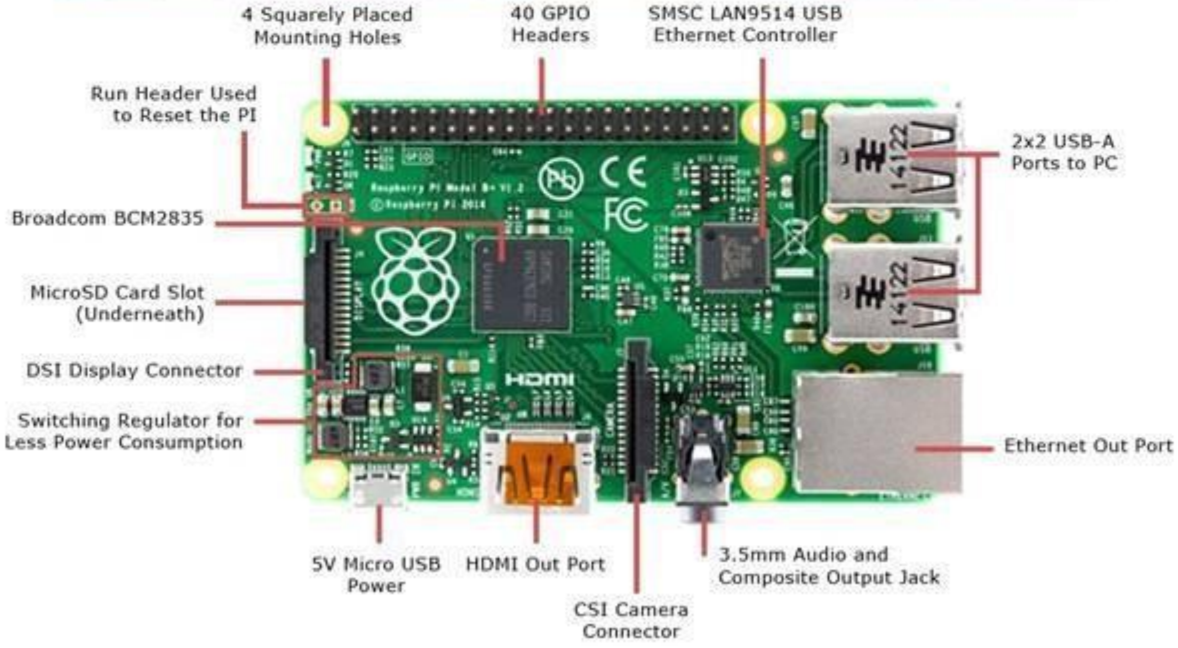


GPIO Pinout Diagram



3V3 Power	1	2	5V Power
GPIO2 SDA1 I2C	3	4	5V Power
GPIO3 SCL1 I2C	5	6	Ground
GPIO4	7	8	GPIO14 UART0_TXD
Ground	9	10	GPIO15 UART0_RXD
GPIO17	11	12	GPIO18 PCM_CLK
GPIO27	13	14	Ground
GPIO22	15	16	GPIO23
3V3 Power	17	18	GPIO24
GPIO10 SPI0_MOSI	19	20	Ground
GPIO9 SPI0_MISO	21	22	GPIO25
GPIO11 SPI0_SCLK	23	24	GPIO8 SPI0_CE0_N
Ground	25	26	GPIO7 SPI0_CE1_N
ID_SD I2C ID EEPROM	27	28	ID_SC I2C ID EEPROM
GPIO5	29	30	Ground
GPIO6	31	32	GPIO12
GPIO13	33	34	Ground
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
Ground	39	40	GPIO21

Pi Model B/B+
Pi Model B+


4 Squarely Placed Mounting Holes
40 GPIO Headers
SMSC LAN9514 USB Ethernet Controller
Run Header Used to Reset the PI
Broadcom BCM2835
MicroSD Card Slot (Underneath)
DSI Display Connector
Switching Regulator for Less Power Consumption
2x2 USB-A Ports to PC
Ethernet Out Port
5V Micro USB Power
HDMI Out Port
CSI Camera Connector
3.5mm Audio and Composite Output Jack

- ***Memory:*** The raspberry pi model Aboard is designed with 256MB of SDRAM and model B is designed with 51MB.Raspberry pi is a small size PC compare with other PCs. The normal PCs RAM memory is available in gigabytes. But in raspberry pi board, the RAM memory is available more than 256MB or 512MB

- ***CPU (Central Processing Unit):*** The Central processing unit is the brain of the raspberry pi board and that is responsible for carrying out the instructions of the computer through logical and mathematical operations. The raspberry pi uses ARM11 series processor, which has joined the ranks of the Samsung galaxy phone.

- ***GPU (Graphics Processing Unit):*** The GPU is a specialized chip in the raspberry pi board and that is designed to speed up the operation of image calculations. This board designed with a Broadcom video core IV and it supports OpenGL
- ***Ethernet Port:*** The Ethernet port of the raspberry pi is the main gateway for communicating with additional devices. The raspberry pi Ethernet port is used to plug your home router to access the internet.

- ***GPIO Pins:*** The general purpose input & output pins are used in the raspberry pi to associate with the other electronic boards. These pins can accept input & output commands based on programming raspberry pi. The raspberry pi affords digital GPIO pins. These pins are used to connect other electronic components. For example, you canconnect it to the temperature sensor to transmit digital data.

- ***XBee Socket:*** The XBee socket is used in raspberry pi board for the wireless communication purpose.

- ***Power Source Connector:*** The power source cable is a small switch, which is placed on side of the shield. The main purpose of the power source connector is to enable an external power source.

- ***UART:*** The Universal Asynchronous Receiver/ Transmitter is a serial input & output port. That can be used to transfer the serial data in the form of text and it is useful for converting the debugging code.

- ***Display:*** The connection options of the raspberry pi board are two types such as HDMI and Composite. Many LCD and HD TV monitors can be attached using an HDMI malecable and with a low-cost adaptor. The versions of HDMI are 1.3 and 1.4 are supported and 1.4 version cable is recommended. The O/Ps of the Raspberry Pi audio and video through HMDI, but does not support HDMI I/p. Older TVs can be connected using composite video. When using a composite video connection, audio is available from the3.5mm jack socket and can be sent to your TV. To send audio to your TV, you need a cable which adjusts from 3.5mm to double RCA connectors.

Arduino

i. Arduino is an open-source hardware and software company that designs and manufactures single-board microcontrollers and microcontroller kits.

ii. In Arduino, each board has clear markings on the connection pins, sockets and in-circuit connections. Thus, Arduino boards are easy to work for DIY (do-it-yourself) and simplify the prototyping of embedded platforms for IoTs.

iii. The Arduino Integrated Development Environment or Arduino Software (IDE) are open source for easy to program.

iv. Uno is most used and documented board of the whole Arduino family at present. The board's analog input pins and PWM pins can connect sensors, actuators and analog circuits. The board's digital I/O pins can connect On-Off states, set of On-Off states, digital inputs from sensors, digital outputs to actuators and other digital circuits.

v. A board with a shield inserted into it makes a wireless connection to a ZigBee, Bluetooth LE, WiFi, GSM, or RF module or a wired connection to Ethernet LAN for the Internet.

vi. Development boards for IoT devices are the Arduino Ethernet, Arduino Wi-Fi and Arduino GSM shields. Development boards for the wearable devices are Arduino Gemma, LilyPad, LilyPad Simple/SimpleSnap and LilyPad USB.

Figure 8.2 shows architecture of Arduino Fio board with Ethernet shield.

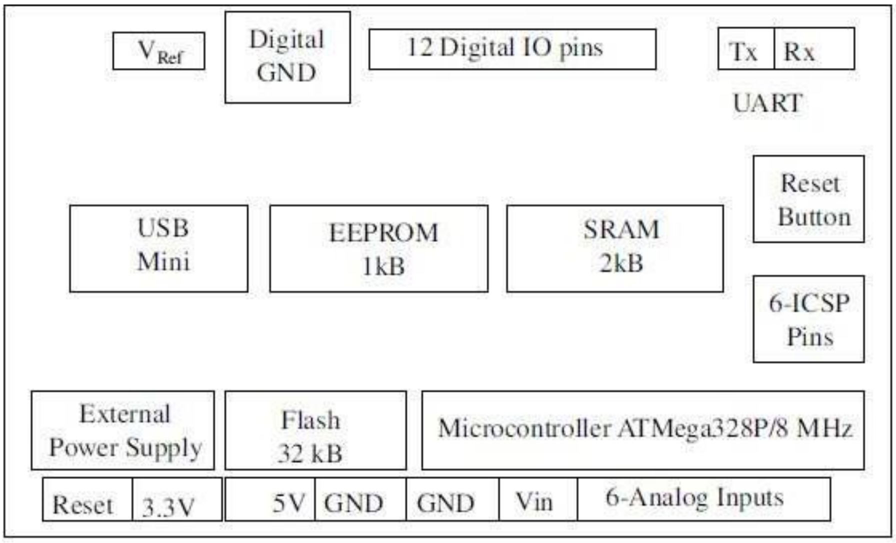


Figure 2: Architecture of Arduino Fio board ford IoT devices development

Arduino types

Arduino Uno (R3), Arduino Nano, Arduino Micro, Arduino Due, LilyPad Arduino Board, Arduino Bluetooth, Arduino Fio, Arduino Diecimila, RedBoard Arduino Board, Arduino Mega (R3) Board Arduino Leonardo Board, Arduino Robot, Arduino Esplora, Arduino Pro Mic Arduino Ethernet, Arduino Zero, Fastest Arduino Board

https://www.elprocus.com/different-types-of-arduino-boards/

https://www.elprocus.com/different-types-of-arduino-boards/

Raspberry Pi

Raspberry Pi is a low-cost mini-computer with the physical size of a credit card. Raspberry Pi runs various flavours of Linux and can perform almost all tasks that a normal desktop computer can do, In addition, Raspberry Pi also allows interfacing sensors and actuators through the general purpose I/O pins. Since Raspberry Pi runs Linux operating system, it supports Python "out of the box".

The different types of raspberry pi models are following
Raspberry Pi 1 model B
Raspberry Pi 1 model A
Raspberry Pi 1 model B+
Raspberry Pi 1model A+
Raspberry Pi Zero
Raspberry Pi 2
Raspberry Pi 3 model B
Raspberry Pi Zero W

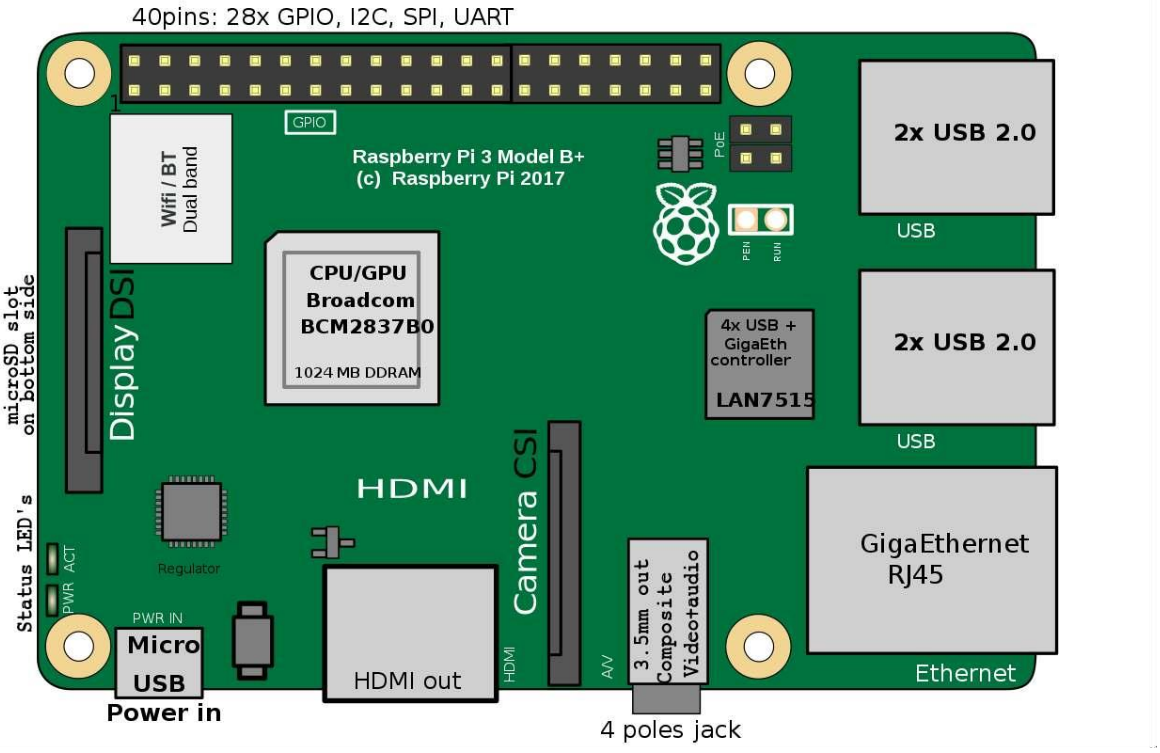


Figure 3: Raspberry Pi Model B+ layout

Types comparison https://www.efxkits.us/different-types-of-raspberry-pi-boards-its-application/
Other information https://en.wikipedia.org/wiki/Raspberry_Pi

i. Raspberry Pi is based on an ARM processor. The latest version of Raspberry Pi (Model B, Revision 2) comes with 700 MHz Low Power ARM 1176JZ-F processor and 512 MB SDRAM,
ii. USB Ports : Raspberry Pi comes with two USB 2.0 ports. The USB ports on Raspberry Pi can provide a current upto 100mA. For connecting devices that draw current more than 100mA an external USB powered hub is required.
iii. Ethernet Ports : Raspberry Pi comes with a standard RJ45 Ethernet port. You can connect an Ethernet cable or a USB Wifi adapter to provide Internet connectivity.
iv. HDMI Output : The HDMI port on Raspberry Pi provides both video and audio output. You can connect the Raspberry Pi to a monitor using an EIDMI cable.
v. Composite Video Output : Raspberry Pi comes with a composite video output.
vi. Audio Output : Raspberry Pi has a 3.5mm audio output jack. The audio quality from this jack is inferior to the HDMI output.
vii. GPIO Pins : Raspberry N comes with a number of general purpose input output pins. There are four types of pins on Raspberry Pi - true GPIO pins. I2C interface pins, SPI interface pins and serial Rx and Tx pins.

a. Serial: The serial interface on Raspberry Pi has receive (Rx) and transmit (Tx) pins for communication with serial peripherals.

b. SPI: Serial Peripheral Interface (SPI) is a synchronous serial data protocol used for communicating with one or more peripheral devices. In an SPI connection, there is one master device and one or more peripheral devices. There are five pins on Raspberry Pi for SPI interface:

- MISO (Master In Slave Out) : Master line for sending data to the peripherals.
- MOSI (Master Out Slave In) : Slave line for sending data to the master.
- SCI (Serial Clock) : Clock generated by master to synchronize data transmission
- CEO (Chip Enable 0) : To enable or disable devices.
- CEO (Chip Enable 1) : To enable or disable devices,

c. I2C: The. 12C interface pins on Raspberry Pi allow you to connect hardware modules. 12C interface allows synchronous data transfer with just two pins - SDA (data line) and SCL (clock line).

viii. Display Serial Interface (1351) The DSI interface can be used to connect an LCD panel to Raspberry Pi.

ix. Camera Serial interface (CSI) : The CSI interface can be used to connect a camera module to Raspberry Pi.

x. Status LEDs : Raspberry Pi has live status LEDs, Table 7.1 lists Raspberry Pi status LEDs and their functions.

xi. SD Card Slot : Raspberry Pi does not have a built in operating system and storage. You can plug-in an SD card loaded with a Linux image to the SD card slot. Appendix-A provides instructions on setting up New Out-of-the-Box Software (NOOBS) on Raspberry Pi. You will require at least an 80H SD card for setting up NOOBS,

xii. Power Input : Raspberry Pi has a micro-USB connector for power input.

ARM Cortex-M class processor

Over the years, ARM has developed quite a number of different processor products. In the following diagram (Figure 4), the ARM processors are divided between the classic ARM processors and the newer Cortex processor product range. In addition, these processors are divided into three groups:

Application Processors – High-end processors for mobile computing, smart phone, servers, etc. These processors run at higher clock frequency (over 1GHz), and support Memory Management Unit (MMU), which is required for full feature OS such as Linux, Android, MS Windows and mobile OSs. If you are planning to develop a product that requires one of these OSs, you need to use an application processor.

Real-time Processors – These are very high-performance processors for real-timeapplications such as hard disk controller, automotive power train and base band control in wireless communications. Most of these processors do not have MMU, and usually have Memory Protection Unit (MPU), cache, and other memory features designed for industrial applications. They can run at a fairly high clock frequency (e.g. 200MHz to >1GHz) and havevery low response latency. Although these processors cannot run full versions of Linux or Windows, there are plenty of Real Time Operating Systems (RTOS) that can be used with these processors.

Microcontroller Processors – These processors are usually designed to have a much lower silicon area, and much high-energy efficiency. Typically, they have shorter pipeline, and usually lower maximum frequency (although you can find some of these processors running at over 200MHz). At the same time, the newer Cortex-M processor family is designed to be very easy to use; therefore, they are very popular in the microcontroller and deeply embedded systems market.

Today, there are eight members in the ARM Cortex-M processor family. Different processors can have different instruction set support, system features and performance.

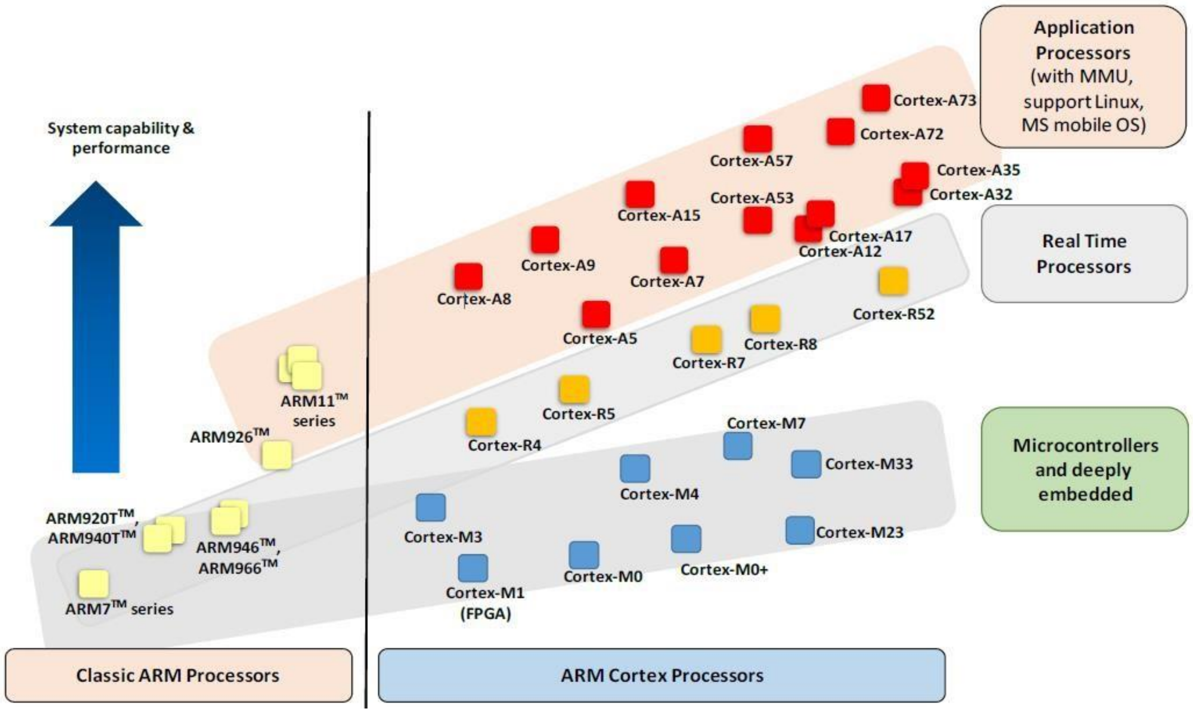


Figure 4: ARM processor family

The Cortex-M processor family

The Cortex-M processor family is more focused on the lower end of the performance scale. However, these processors are still quite powerful when compared to other typical processors used in most microcontrollers. For example, the Cortex-M4 and Cortex-M7 processors are being used in many high-performance microcontroller products, with maximum clock frequency going up to 400MHz. Of course, performance is not the only factor when selecting a processor. In many applications, low power and cost are the key selection criteria. Therefore, the Cortex-M processor family contains various products to address different needs:

Cortex-M0 A very small processor (starting from 12K gates) for low cost, ultra-low power microcontrollers and deeply embedded applications.
Cortex-M0+ The most energy-efficient processor for small embedded system. Similar size and programmer's model to the Cortex-M0 processor, but with additional features like single cycle I/O interface and vector table relocations.

Cortex-M1 A small processor design optimized for FPGA designs and provides Tightly Coupled Memory (TCM) implementation using memory blocks on the FPGAs. Same instruction set as the Cortex-M0.

Cortex-M3 A small but powerful embedded processor for low-power microcontrollers that has a rich instruction set to enable it to handle complex tasks quicker. It has a hardware divider and Multiply-Accumulate (MAC) instructions. In addition, it also has comprehensive debug and trace features to enable software developers to develop their applications quicker.

Cortex-M4 It provides all the features on the Cortex-M3, with additional instructions target at Digital Signal Processing (DSP) tasks, such as Single Instruction Multiple Data (SIMD) and faster single cycle MAC operations. In addition, it also have an optional single precision floating point unit that support IEEE 754 floating point standard.

Cortex-M7 High-performance processor for high-end microcontrollers and processing intensive applications. It has all the ISA features available in Cortex-M4, with additional support for double-precision floating point, as well as additional memory features like cache and Tightly Coupled Memory (TCM).

Arm Cortex-M0 Processor Architecture

The ARMv6-M architecture that the Cortex-M0 processor implemented covers a number of different areas. To use a Cortex-M0 device with C language, you only need to know the memory map, the peripheral programming information, the exception handling mechanism, and part of the programmer's model. Most users of the Cortex-M0 processor will work in C language; as a result, the underlying programmer's model will not be visible in the program code. However, it is still useful to know about the details, as this information is often needed during debugging and it will also help readers to understand the rest of this book.

Programmer's Model

1. Operation Modes and States

i. The Cortex-M0 processor has two operation modes (Thread mode or the Handler mode) and two states (Thumb and Deburg stated) as in the Figure 5.

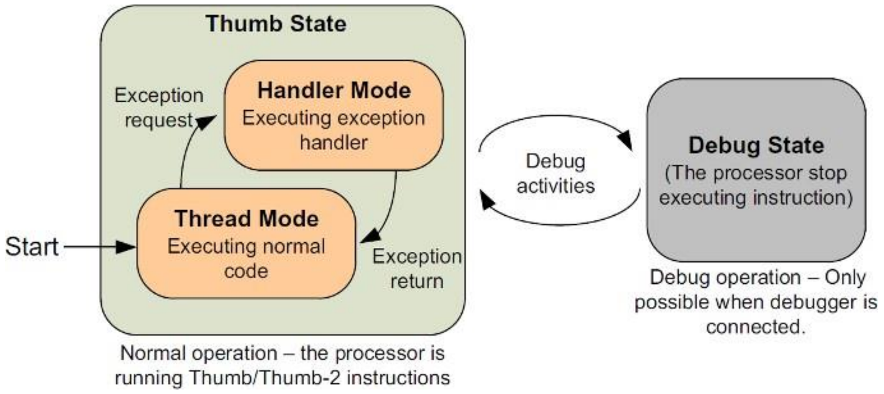

Thumb State
Exception request
Handler Mode
Executing exception handler
Thread Mode
Executing normal code
Exception return
Start
Debug activities
Debug State
(The processor stop executing instruction)
Debug operation – Only possible when debugger is connected.
Normal operation – the processor is running Thumb/Thumb-2 instructions

Figure 5: Processor modes and states in the Cortex-M0 processor

ii. When the processor is running a program, it is in the Thumb state. In this state, it can be either in the Thread mode or the Handler mode. Both modes are almost the same. The only difference is Thread mode have a special register called CONTROL.

iii. The Debug state is used for debugging operation only. Halting the processor, stops the instruction execution and enter debug state. This state allows the debugger to access or change the processor register values.

iv. The debugger can access system memory locations in either the Thumb state or the Debug state.

v. When the processor is powered up, it will be running in the Thumb state and Thread mode by default.

2. Registers and Special Registers

i. To perform data processing and controls, a number of registers are required inside the processor core. The data have to be loaded from the memory to a register in the register bank then processed inside the processor, and then written back to the memory if needed. This is commonly called as “load-store architecture.”

ii. By having a sufficient number of registers in the register bank, this mechanism is easy to use. The register bank contains sixteen 32-bit registers. 13 are general-purpose registers, remaining have special uses as shown in the Figure 6.

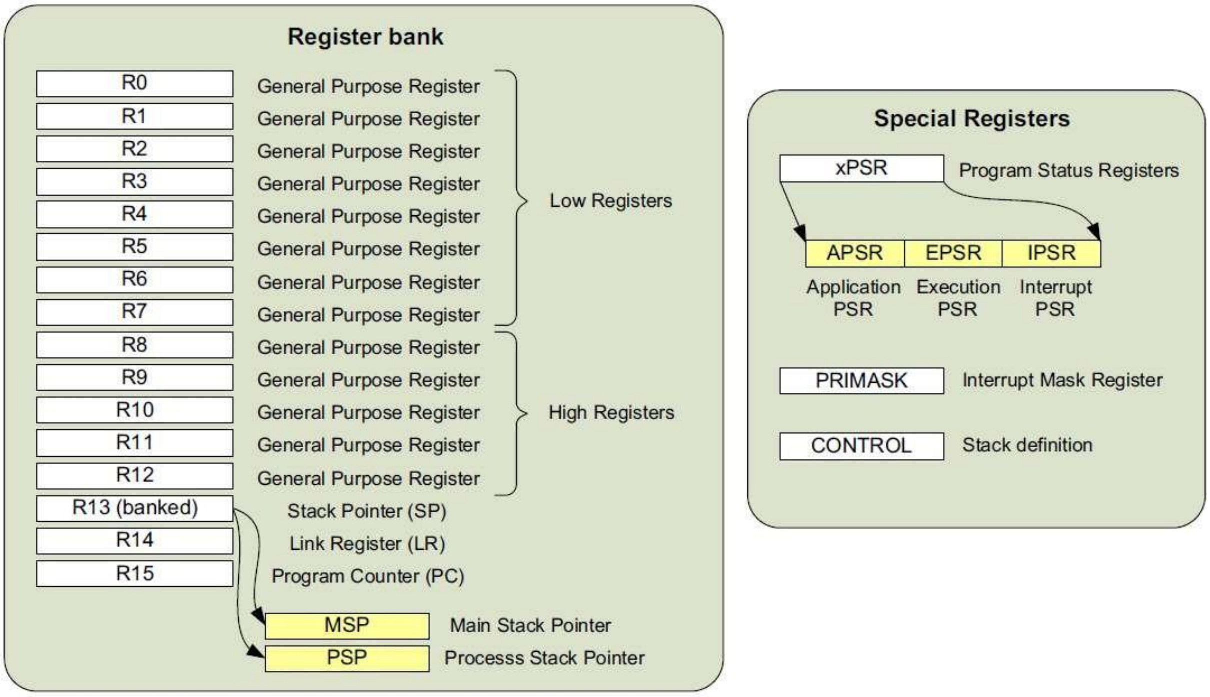


Figure 6: Registers in the Cortex-M0 processor

R0-R12: Registers R0 to R12 are for general uses. The Thumb instructions can only access low registers (R0 to R7). Some instructions like MOV (move) can use all registers. The initial values of R0 to R12 at reset are undefined.

R13, Stack Pointer (SP): R13 is the stack pointer. It is used for accessing the stack memory via PUSH and POP operations. There are physically two different stack pointers in Cortex-M0. The main stack pointer (MSP) is used for running unusual handlers mode and process stack pointer (PSP) is used for usual Thread mode.

R14, Link Register (LR): R14 is the Link Register. The Link Register is used for storing the return address of a function call.

R15, Program Counter (PC): R15 is the Program Counter. It is readable and writeable. Call the Program Counter, using either “R15” or “PC,” in either upper or lower case (e.g., “r15” or “pc”).

xPSR, combined Program Status Register

The combined Program Status Register provides information about program execution and the ALU flags. It is consists of the following three Program Status Registers (PSRs) as in Figure 7:

- Application PSR (APSR): Contains the ALU flags: N (negative flag), Z (zero flag), C (carry or borrow flag), and V (overflow flag). These bits are at the top 4 bits of the APSR.
- Interrupt PSR (IPSR): Contains the current executing interrupt service routine (ISR) number.
- Execution PSR (EPSR): Contains the T-bit, which indicates that the processor is in the Thumb state.

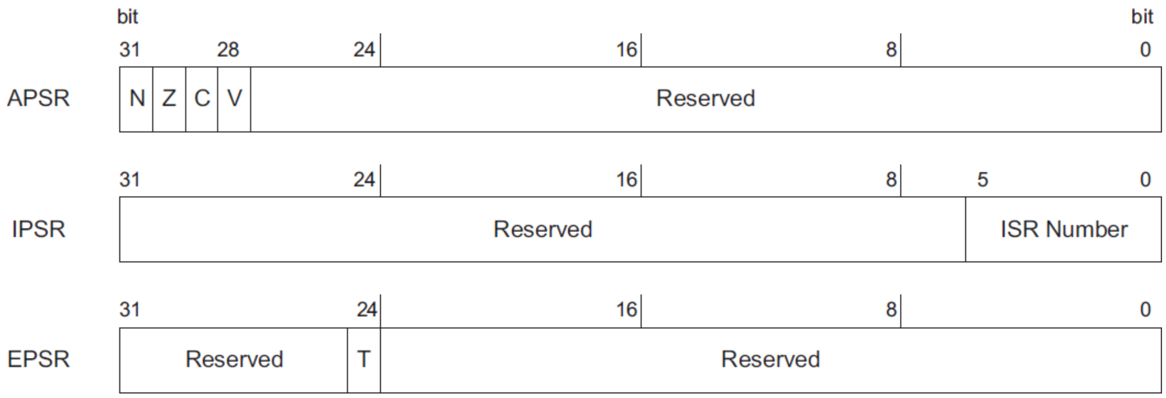


Figure 7: APSR, IPSR, and EPSR.

These three registers can be accessed as one register called xPSR as given in Figure 8.

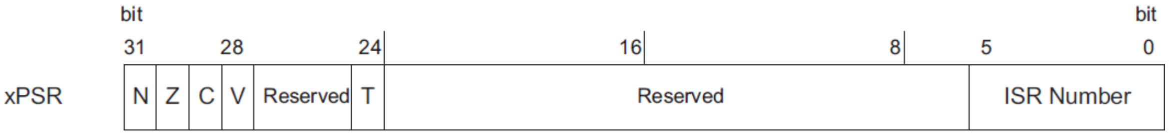


Figure 8: xPSR

PRIMASK: Interrupt Mask Special Register

The PRIMASK register is a 1-bit-wide interrupt mask register as in Figure 9. When set, it blocks all interrupts apart from the non-maskable interrupt (NMI) and the hard fault exception. Effectively it raises the current interrupt priority level to 0, which is the highest value for a programmable exception. The PRIMASK register can be accessed using special register access instructions (MSR, MRS) as well as using an instruction called the Change Processor State (CPS). This is commonly used for handling time-critical routines.

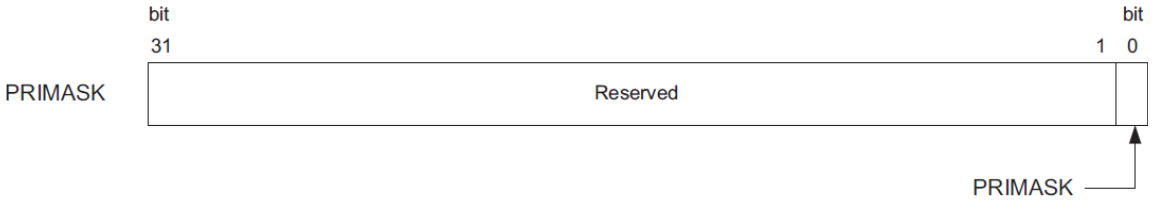


Figure 9: PRIMASK

CONTROL: Special Register

i. Physically there are two stack pointers in the Cortex-M0 processor, but only one of them is used at one time, depending on the current value of the CONTROL register as shown in the Figure 10.

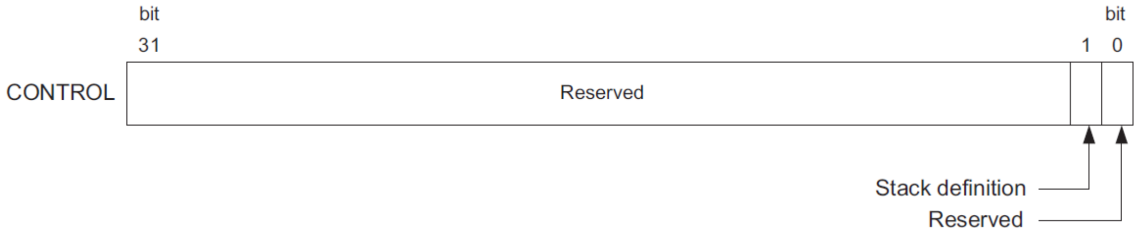


Figure 10: CONTROL

ii. After reset, the main stack pointer (MSP) is used, but can be switched to the process stack pointer (PSP) in Thread mode by setting bit [1] in the CONTROL register as shown in the Figure 11.

iii. During running of an exception handler (when the processor is in Handler mode), only the MSP is used, and the CONTROL register reads as zero.

iv. Bit 0 of the CONTROL register is reserved to maintain compatibility with the Cortex-M3 processor.

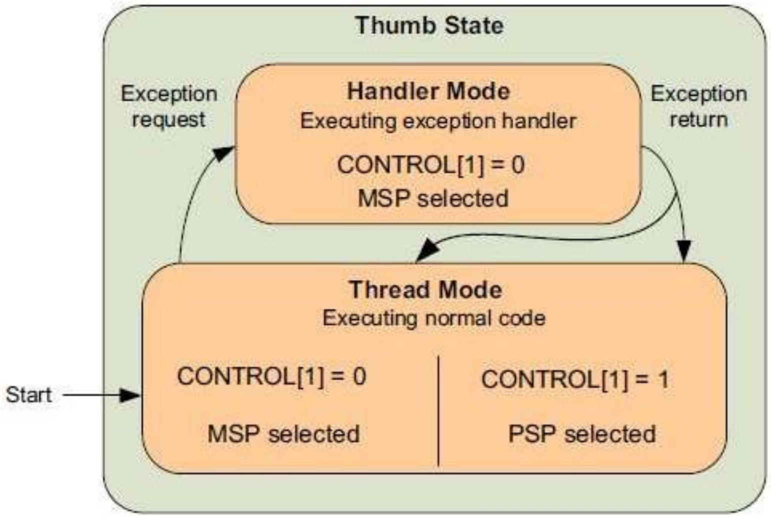


Figure 11: Stack pointer selection

3. Memory System Overview

The Cortex-M0 processor has 4 GB of memory address space. The memory space is architecturally defined as a number of regions, with each region having a recommended usage to help software porting between different devices as shown in the Figure 12.

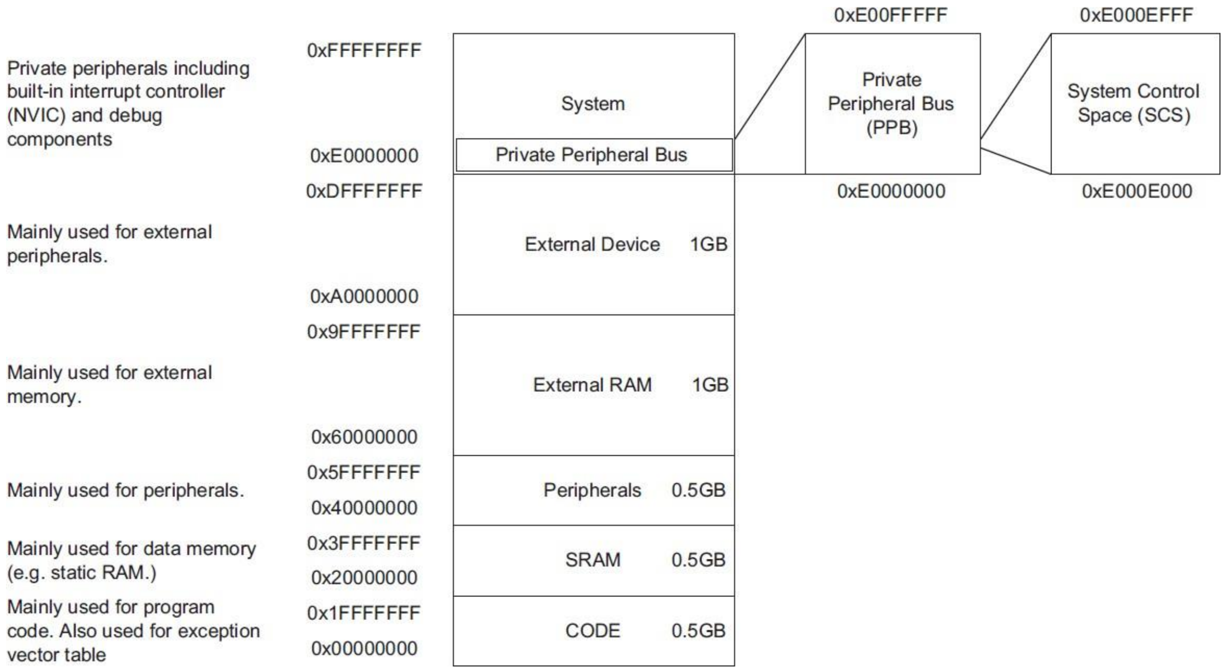

0xE00FFFFF
0xE000EFFF
0xFFFFFFFF
Private peripherals including built-in interrupt controller (NVIC) and debug components
System
Private Peripheral Bus (PPB)
System Control Space (SCS)
0xE0000000
Private Peripheral Bus
0xDFFFFFFF
0xE0000000
0xE000E000
Mainly used for external peripherals.
External Device 1GB
0xA0000000
0x9FFFFFFF
Mainly used for external memory.
External RAM 1GB
0x60000000
0x5FFFFFFF
Mainly used for peripherals.
Peripherals 0.5GB
0x40000000
0x3FFFFFFF
Mainly used for data memory (e.g. static RAM.)
SRAM 0.5GB
0x20000000
Mainly used for program code. Also used for exception vector table
0x1FFFFFFF
CODE 0.5GB
0x00000000

Figure 12: Memory map

Stack Memory Operations

Stack memory is a memory usage mechanism that allows the system memory to be used as temporary data storage. The main element of stack memory operation is a register called the stack pointer. The stack pointer is adjusted automatically each time a stack operation is carried out. In common terms, storing data to the stack is called pushing (using the PUSH instruction) and restoring data from the stack is called popping (using the POP instruction) as shown in the Figure 13.

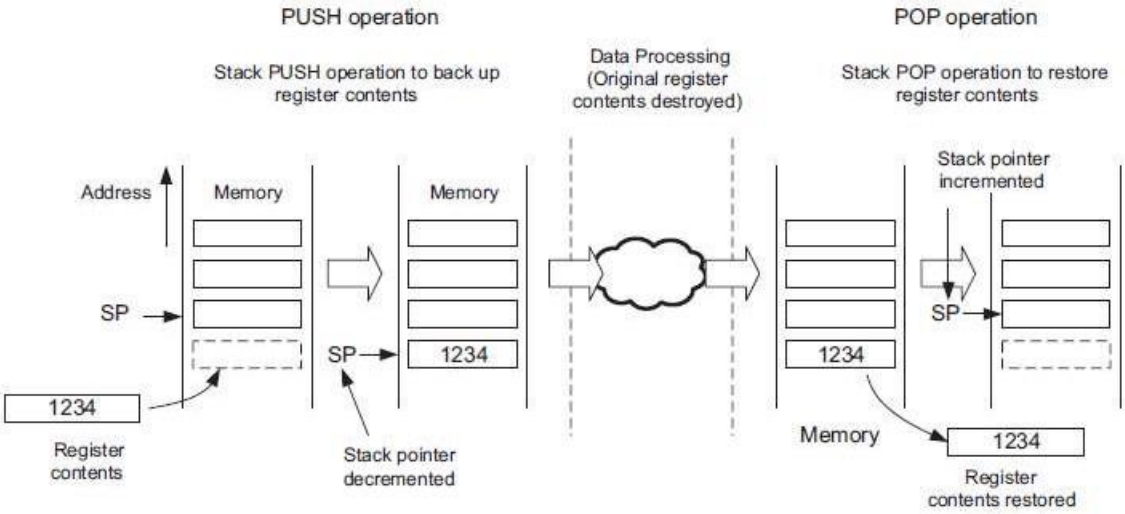

PUSH operation
Stack PUSH operation to back up register contents
Data Processing (Original register contents destroyed)
POP operation
Stack POP operation to restore register contents
Address
Memory
Memory
Stack pointer incremented
SP
SP
SP
1234
1234
1234
Register contents
Stack pointer decremented
Memory
1234
Register contents restored

Figure 13: Stack PUSH and POP in the Cortex-M0 processor.

4. Exceptions and Interrupts

i. Exceptions are events that cause change to program control: instead of continuing program execution, the processor suspends the current executing task and executes a part of the program code called the exception handler. After the exception handler is completed, it will then resume the normal program execution.

ii. There are various types of exceptions. Interrupts are a subset of exceptions. They are 32 external interrupts (commonly referred as interrupt request, IRQs) and an additional special interrupt called the nonmaskable interrupt (NMI).

iii. The exception handlers for interrupt events are commonly known as interrupt service routines (ISRs).

Nested Vectored Interrupt Controller (NVIC)

To prioritize the interrupt requests and handle other exceptions, the Cortex-M0 processor has a built-in interrupt controller called the Nested Vectored Interrupt Controller (NVIC). The interrupt management function is controlled by a number of programmable registers in the NVIC. These registers are memory mapped, with the addresses located within the System Control Space (SCS) as illustrated in Figure 12.

The NVIC supports a number of features:

- Flexible interrupt management: each external interrupt can be enabled or disabled. It can also accept exception requests at external peripheral, at 1cycle.
- Nested interrupt support: each exception has a priority level. The priority level can be fixed or programmable.
- Vectored exception entry: When an exception occurs, the processor will need to locate the starting point of the corresponding exception handler.
- Interrupt masking

Interrupt Masking

The NVIC in the Cortex-M0 processor provides an interrupt masking feature via the PRIMASK special register. This can disable all exceptions except hard fault and NMI (non- maskable interrupt, ~~hardware failure~~). This masking is useful for operations that should notbe interrupted such as time critical control tasks.

5. System Control Block (SCB)

Apart from the NVIC, the System Control Space (SCS) also contains a number of other registers for system management. This is called the System Control Block (SCB). It contains registers for sleep mode features and system exception configurations, as well as a register containing the processor identification code.

Block Diagram

i. A simplified block diagram of the Cortex-M0 is shown in Figure 14. The processor core contains the register banks, ALU, data path, and control logic. It is a three stage pipeline design with fetch stage, decode stage, and execution stage. The register bank has sixteen 32- bit registers. A few registers have special usages.

ii. The Nested Vectored Interrupt Controller (NVIC) accepts up to 32 interrupt request signals and a non-maskable interrupt (NMI) input.

iii. It contains the functionality required for comparing priority between interrupt requests and the current priority level so that nested interrupts can be handled automatically.

iv. If an interrupt is accepted, it communicates with the processor so that the processor can execute the correct interrupt handler. The Wakeup Interrupt Controller (WIC) is an optional unit.

iv. In low-power applications, the microcontroller can enter standby state with most of the processor powered down. In this situation, the WIC can perform the function of interrupt masking.

v. When an interrupt request is detected, the WIC informs the power management to power up the system so that the NVIC and the processor core can then handle the rest of the interrupt processing. The debug subsystem contains various functional blocks to handledebug control, program breakpoints, and data watch points.

vi. The serial wire protocol is a newer communication protocol that only requires two wires, but it can handle the same debug functionalities as JTAG (Joint Test Action Group). The internal bus system, the data path in the processor core, and the AHB LITE bus interface are all 32 bits wide.

vii. AHB-Lite is an on-chip bus protocol used in many ARM processors. This bus protocol is part of the Advanced Microcontroller Bus Architecture (AMBA) specification, a bus architecture developed by ARM that is widely used in the IC design industry.

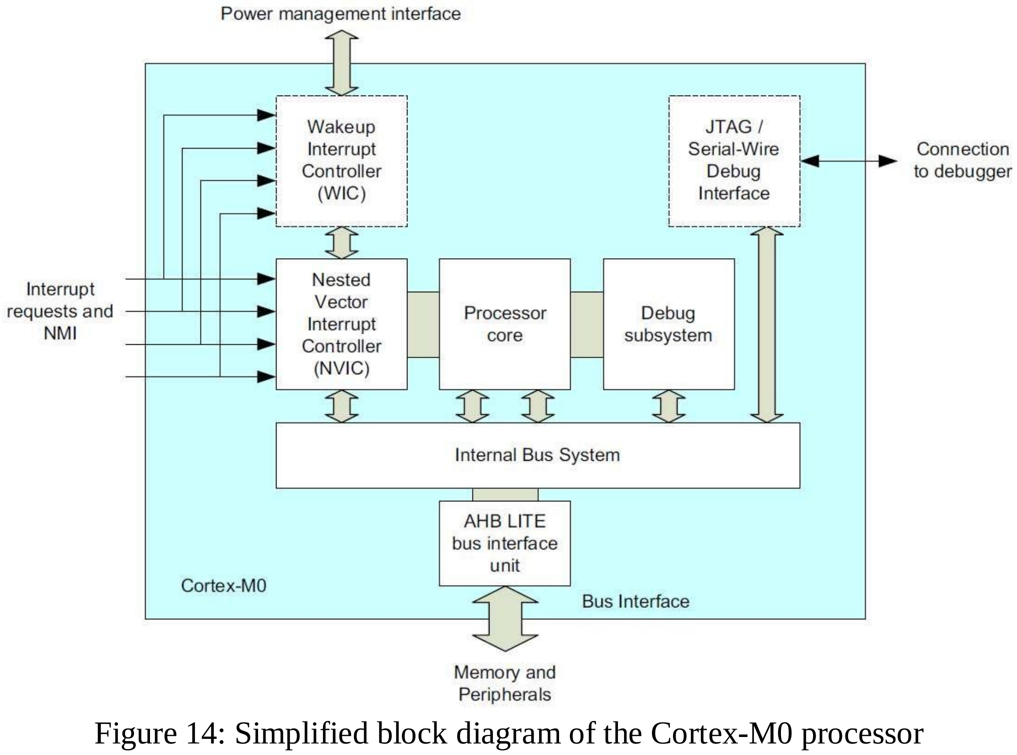

Power management interface
Wakeup Interrupt Controller (WIC)
JTAG / Serial-Wire Debug Interface
Connection to debugger
Interrupt requests and NMI
Nested Vector Interrupt Controller (NVIC)
Processor core
Debug subsystem
Internal Bus System
AHB LITE bus interface unit
Cortex-M0
Bus Interface
Memory and Peripherals

Figure 14: Simplified block diagram of the Cortex-M0 processor

CHAPTER 5

Instruction Set

Background ofARM and Thumb Instruction Set

The ealy ARM processors use a 32-bit instruction set called the ARM instructions. The 32-bit ARM instruction set is powerfiul and provides good perfiormance, but at the same time it often requires larger program memory when compared to 8-bit and 16-bit processors. This was and still is an issue, as memory is expensive and could consume a considerable amount of power.

In 1995, ARM introduced the ARM7TDMI processor, adding a new 16-bit instruction set called the Thumb instruction set. The ARM7TDMI supports both ARM instructions and Thumb instructions, and a state-switching mechanism is used to allow the processor to decide which instruction decode scheme should be used (Figure 5.1). The Thumb instruction set provides a subset of the ARM instructions. By itself it can perform most of the normal functions, but interrupt entry sequence and boot code must still be in ARM state. Nevertheless, most processing can be carried out using Thumb instructions and interrupt handlers could switch themselves to use the Thumb state, so the ARM7TDMI processor provides excellent code density when compared to other 32-bit RISC architectures.

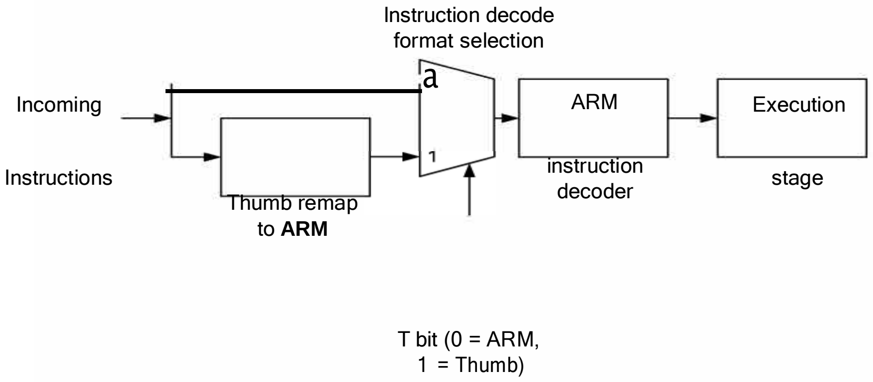


Figure 5.1:
ARM7TDMI design supports both ARM and the Thumb instruction set.

Thumb code provides a code size reduction of approximately 30% compared to the equivalent ARM code. However, it has some impact on the perfiormance and can reduce the perfiormanceby 20%. On the other hand, in many applications, the reduction of program memory size and

The Definitive Guide to the ARM Cortex-MO. DOI: 10.1016/B978-0-12-385477-3.10005-9

the low-power nature of the ARM7TDMI processor made it extremely popular with portable electronic devices like mobile phones and microcontrollers.

In 2003, ARM introduced Thumb-2 technology. This technology provides a number of 32-bit Thumb instructions as well as the original 16-bit Thumb instructions. The new 32-bit Thumb instructions can carry out most operations that previously could only be done with the ARM instruction set. As a result, program code compiled for Thumb-2 is typically 74% of the size of the same code compiled for ARM, but it maintains similar performance.

The Cortex-M3 processor is the first ARM processor that supports only Thumb-2 instructions. It can deliver up to 1.25 DMIPS per MHz (measured with Dhrystone 2.1), and various microcontroller vendors are already shipping microcontroller products based on the Cortex-M3 processor. By implementing only one instruction set, the software development is made simpler and at the same time improves the energy efficiency because only one instruction decoder is required (Figure 5.2).

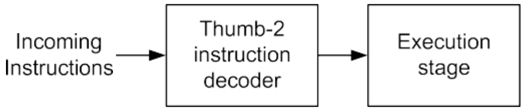


Figure 5.2:
Cortex-M processors do not have to remap instructions from Thumb to ARM.

In the ARMv6-M architecture used in the Cortex-M0 processor, in order to reduce the circuit size to a minimum, only the 16-bit Thumb instructions and a minimum subset of 32-bit Thumb instructions are supported. These 32-bit Thumb instructions are essential because the ARMv6-M architecture uses a number of features in the ARMv7-M architecture, which requires these instructions. For example, the accesses to the special registers require the MSR and MRS instructions. In addition, the Thumb-2 version of Branch and Link instruction (BL) is also included to provide a larger branch range.

Although the Cortex-M0 processor does not support many 32-bit Thumb instructions, the Thumb instruction set used in the Cortex-M0 processor is a superset of the original 16-bit Thumb instructions supported on the ARM7TDMI, which is based on ARMv4T architecture. Over the years, both ARM and Thumb instructions have gone through a number of enhancements as the architecture has evolved. For example, a number of instructions for data type conversions have been added to the Thumb instruction set for the ARMv6 and ARMv6-M architectures. These instruction set enhancements, along with various implementation optimizations, allow the Cortex-M0 processor to deliver the same level of performance as an ARM7TDMI running ARM instructions.

Table 5.1 shows the base 16-bit Thumb instructions supported in the Cortex-M0.

Table 5.1: 16-Bit Thumb Instructions Supported on the Cortex-M0 Processor

16-Bit Thumb Instructions Supported on Cortex-M0									
ADC	ADD	ADR	AND	ASR	B	BIC	BLX	BKPT	BX
CMN	CMP	CPS	EOR	LDM	LDR	LDRH	LDRSH	LDRB	LDRSB
LSL	LSR	MOV	MVN	MUL	NOP	ORR	POP	PUSH	REV
REV16	REVSH	ROR	RSB	SBC	SEV	STM	STR	STRH	STRB
SUB	SVC	SXTB	SXTH	TST	UXTB	UXTH	WFE	WFI	YIELD

The Cortex-M0 processor also supports a number of 32-bit Thumb instructions from Thumb-2 technology (Table 5.2):

- MRS and MSR special register access instructions
- ISB, DSB, and DMB memory synchronization instructions
- BL instruction (BL was supported in traditional Thumb instruction set, but the bit field definition was extended in Thumb-2)

Table 5.2: 32-Bit Thumb Instructions Supported on the Cortex-M0 Processor

32-Bit Thumb Instructions Supported on Cortex-M0					
BL	DSB	DMB	ISB	MRS	MSR

Assembly Basics

This chapter introduces the instruction set of the Cortex-M0 processor. In most situations,