International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 06 Issue 13
December 2019

IOT Based Data Acquisition System

B Varalakshmi[1], M.Anil Kumar[2], S.Mahaboob Basha[3]
[1]P.G. Scholar, [2] Guide, Assistant Professor, [3]Head of the Department
[1,2,3]Branch:Embedded systems, ECE, M.Tech
[1.2,3] Electronics and Communication Engineering Department
[1,2 ,3]Geethanjali College of Engineering &Technology, Kurnool
Email.Id:[1]varalakshmi16510@gmail.com, [2]anilkumarmadhavaram907@gmail.com

Abstract*:*

The development in wireless sensor networks can be used in monitoring and controlling various parameters in the agriculture field, weather station field. Due to uneven and natural distribution of rain water it is very difficult for farmers to monitor and control the distribution of water to agriculture field in the whole farm or as per the requirement of the crop. There is no ideal and advanced irrigation method for all weather conditions, soil structure and variety of crops cultures. Farmers suffer large financial losses because of wrong prediction of weather and incorrect irrigation methods and the amount of pesticides and insecticides used for crops. In this context, with the evolution of miniaturized sensor devices coupled with wireless technologies, it is possible remotely monitor parameters such as temperature and humidity and sun light intensity. Weather monitoring plays an important role in human life, so the collection of information about the temporal dynamics of weather.

Keywords:- — Data Acquisition System, Raspberrry Pi, IOT, Ethernet, Wi-Fi, Sensors.

INTRODUCTION

The development in wireless sensor networks can be used in monitoring and controlling various parameters in the agriculture field, weather station field. Due to uneven and natural distribution of rain water it is very difficult for farmers to monitor and control the distribution of water to agriculture field in the whole farm or as per the requirement of the crop. There is no ideal and advanced irrigation method for all weather conditions, soil structure and variety of crops cultures. Farmers suffer large financial losses because of wrong prediction of weather and incorrect irrigation methods and the amount of pesticides and insecticides used for crops. In this context, with the evolution of miniaturized sensor devices coupled with wireless technologies, it is possible remotely monitor parameters such as temperature and humidity and sun light intensity. Weather monitoring plays an important role in human life, so the collection of information about the temporal dynamics of weather.

Literature Survey

Arko Djajad et al. have presented their system for ambient environmental quality monitoring using IoT sensor network. In this system sensors are connected to Net Client via serial interfaces such as Modbus or I2C. Data collection is then sent to Fognet by TCP/IP. Data from all environmental sensors act as input to Arduino ATMega2560 board. This board needs Wi-Fi module to connect to Wi-Fi router. This result can be sent to the web-server to achieve online monitoring purpose. Result can be accessed by

electronic devices such as laptop or hand phone as long as it is connected to internet. Sensors used are ambient light sensor, DHT11 for temperature and humidity sensor, sound sensor, sensor MG811 for carbon dioxide, sensor MQ7 for carbon monoxide, sensor MQ6 for LPG and sensor MQ3 for alcohol. All environmental sensors used are analog sensors. Hence, these sensors can be connected to analog pin of Arduino. In this research two communications are used, Serial0 for communication between Arduino and PC, and Serial1 for communication between Arduino and ESP8266.

Tamilarasi B. et al. have proposed the system that deals with the functional design and implementation of WSN platform, which can be used for long-term environmental monitoring in IoT application.

Nikhil Ugale et al. have proposed the system based on IoT for environmental condition monitoring in homes. The proposed system use different sensors light, temperature, level and humidity to collect the data to monitor the environmental conditions and also to detect any fault in devices. PIC microcontroller is used to control and monitor all the sensors connected to it. Different sensors are placed against different devices to check the functionality of the devices. Once device turn on sensor will sense the functionality of the device if it is not working properly then automatically e-mail or SMS will send to the service provider as well as user. Also one more functionality is added that device can turn on and off through cell phone as well as to get full functionality details of device through SMS.

Ms. Padwal S. C. et al. have proposed the system to implement a WSN platform that can be used for a range of longterm environmental monitoring for IoT applications. This paper presents functional design of WSN for IoT a2pplication.

HARDWARE IMPLEMENTATION OF THE PROJECT

This chapter briefly explains about the Hardware Implementation of the project. It discusses the design and working of the design with the help of block diagram and circuit diagram and explanation of circuit diagram in detail. It explains the features, timer programming, serial communication, interrupts of Raspberry PI. It also explains the various modules used in this project.

Project Design

The implementation of the project design can be divided in two parts.

- Hardware implementation
- Firmware implementation

Hardware implementation deals in drawing the schematic on the plane paper according to the application, testing the schematic design over the breadboard using the various IC's to find if the design meets the objective, carrying out the PCB layout of the schematic tested on breadboard, finally preparing the board and testing the designed hardware.

The project design and principle are explained in this chapter using the block diagram and circuit diagram. The block diagram discusses about the required components of the design and working condition is explained using circuit diagram and system wiring diagram.

Introduction to Microcontroller

Based on the Processor side Embedded Systems is mainly divided into 3 types

1. Micro Processor : - are for general purpose eg: our personal computer

2. Micro Controller:- are for specific applications, because of cheaper cost we will go for these

3. DSP (Digital Signal Processor):- are for high and sensitive application purpose

Microcontroller versus Microprocessor

A system designer using a general-purpose microprocessor such as the Pentium or the 68040 must add RAM, ROM, I/O ports, and timers externally to make them functional. Although the addition of external RAM, ROM, and I/O ports makes these systems bulkier and much more expensive, they have the advantage of versatility such that the designer can decide on the amount of RAM, ROM and I/O ports needed to fit the task at hand.

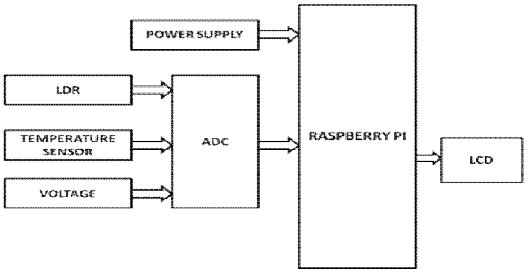
A Microcontroller has a CPU (a microprocessor) in addition to a fixed amount of RAM, ROM, I/O ports, and a timer all on a single chip. In other words, the processor, the RAM, ROM, I/O ports and the timer are all embedded together on one chip; therefore, the designer cannot add any external memory, I/O ports, or timer to it. The fixed amount of on-chip ROM, RAM, and number of I/O ports in Microcontrollers makes them ideal for many applications in which cost and space are critical.

Table: Microprocessor vs. Microcontroller

Microprocessor	Microcontroller
CPU is stand alone RAM, RAM, I/O, timer are separate	CPU,RAM,ROM,I/O and timer are all on a single chip
Designer can decide on the amount of ROM, RAM and I/O ports.	Fix amount of on chip ROM,RAM, I/O Ports.
Expansive, Versatility	For applications in which cost, power and space are critical
General purpose	Single purpose

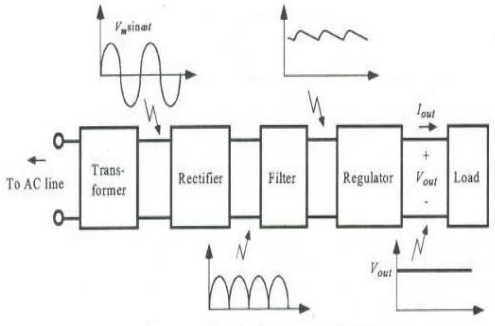
Block Diagram of the Project

The block diagram of the design is as shown in the below fig. It consists of power supply unit, Raspberry pi3 board, buzzer, sensors and ADC MCP3202.



Power Supply:

The input to the circuit is applied from the regulated power supply. The a.c. input i.e., 230V from the mains supply is step down by the transformer to 12V and is fed to a rectifier. The output obtained from the rectifier is a pulsating d.c voltage. So in order to get a pure d.c voltage, the output voltage from the rectifier is fed to a filter to remove any a.c components present even after rectification. Now, this voltage is given to a voltage regulator to obtain a pure constant dc voltage.



Components of a regulated power supply

Transformer:

Usually, DC voltages are required to operate various electronic equipment and these voltages are 5V, 9V or 12V. But these voltages cannot be obtained directly. Thus the a.c input available at the mains supply i.e., 230V is to be brought down to the required voltage level. This is done by a transformer. Thus, a step down transformer is employed to decrease the voltage to a required level.

Rectifier:

The output from the transformer is fed to the rectifier. It converts A.C. into

pulsating D.C. The rectifier may be a half wave or a full wave rectifier. In this project, a bridge rectifier is used because of its merits like good stability and full wave rectification.

Filter:

Capacitive filter is used in this project. It removes the ripples from the output of rectifier and smoothens the D.C. Output received from this filter is constant until the mains voltage and load is maintained constant. However, if either of the two is varied, D.C. voltage received at this point changes. Therefore a regulator is applied at the output stage.

Voltage regulator:

As the name itself implies, it regulates the input applied to it. A voltage regulator is an electrical regulator designed to automatically maintain a constant voltage level. In this project, power supply of 5V and 12V are required. In order to obtain these voltage levels, 7805 and 7812 voltage regulators are to be used. The first number 78 represents positive supply and the numbers 05, 12 represent the required output voltage levels.

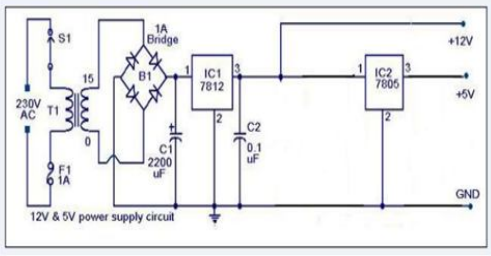


Fig: Power supply circuit diagram

RASPBERRY PI

The Raspberry Pi is a credit-card sized computer that plugs into your TV and a keyboard. It is a capable little computer which can be used in electronics projects, and for many of the things that your desktop PC does, like spreadsheets, word-processing and games. It also plays high-definition video. We want to see it being used by kids all over the world to learn how computers work, how to manipulate the electronic world around them, and how to program.

The Raspberry Pi is a low cost, **credit-card sized computer** that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

What's more, the Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. We want to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work. There are currently four Raspberry Pi models. They are the Model A, the Model B, theModel B+ and the Compute Module. All models use the same CPU, the BCM2835, but other hardware features differ.

THE MODEL B+

Released in July 2014, the Model B+ is a updated revision of the Model B. It increases the number of USB ports to 4 and the number of pins on the GPIO header to 40. In addition, it has improved power circuitry which allows higher powered USB devices to be attached and now hotplugged. The full size composite video connector has been removed and the functionality moved to the 3.5mm audio/video jack. The full size

SD card slot has also been replaced with a much more robust microSD slot.
The following list details some of the improvements over the Model B.

- Current monitors on the USB ports mean the B+ now supports hot plugging.
- Current limiter on the 5V for HDMI means HDMI cable powered VGA converters will now all work
- 14 more GPIO pins
- EEPROM readout support for the new HAT expansion boards
- Higher drive capacity for analog audio out, from a separate regulator, which means a better audio DAC quality.
- No more backpowering problems, due to the USB current limiters which also inhibit back flow, together with the "ideal power diode"
- Composite output moved to 3.5mm jack
- Connectors now moved to two sides of the board rather than the four of the original device.
- Ethernet LED's moved to the ethernet connector
- 4 squarely positioned mounting holes for more rigid attachment to cases etc.

The power circuit changes also means a reduction in power requirements of between 0.5W and 1W.

Product Description:

The Raspberry Pi Model B+ incorporates a number of enhancements and new features. Improved power consumption, increased connectivity and greater IO are among the improvements to thisPowerful, small and lightweight ARM based computer.

THE COMPUTE MODULE

The compute module is intended for industrial applications, it is a cut down device which simply include the BCM2835, 512MB of SDRAM and a 4GB eMMC flash memory, in a small form factor. This connects to a base board using a repurposed 200 pin DDR2 SODIMM connector. Note the device is NOT SODIMM compatible, it just repurposes the connector. All the BCM2835 features are exposed via the SODIMM connector, including twin camera and LCD ports, whilst the Model A or B/B+ only have one of each.

The compute module is expected to be used by companies wishing to shortcut the development process of new product, meaning only a baseboard needs to be developed, with appropriate peripherals, with the Compute Module providing the CPU, memory and storage along with tested and reliable software.

1. BCM2835

The Broadcom chip used in the Raspberry Pi Model A, B and B+. The BCM2835 is a cost-optimized, full HD, multimedia applications processor for advanced mobile and embedded applications that require the highest levels of multimedia performance. Designed and optimized for power efficiency, BCM2835 uses Broadcom's VideoCore® IV technology to enable applications in media playback, imaging, camcorder, streaming media, graphics and 3D gaming.

2. General Purpose I/O (GPIO)

General Purpose Input/Output pins on the Raspberry Pi

OVERVIEW

This page expands on the technical features of the GPIO pins available on BCM2835 in general. For usage examples, see the GPIO Usage section. When reading this page, reference should be made to the BCM2835 ARM Peripherals Datasheet, section 6.
GPIO pins can be configured as either general-purpose input, general-purpose output or as one of up to 6 special alternate settings, the functions of which are pin-dependant.
There are 3 GPIO banks on BCM2835.
Each of the 3 banks has its own VDD input pin. On Raspberry Pi, all GPIO banks are

supplied from 3.3V. **Connection of a GPIO to a voltage higher than 3.3V will likely destroy the GPIO block within the SoC.**
A selection of pins from Bank 0 is available on the P1 header on Raspberry Pi.

GPIO PADS

The GPIO connections on the BCM2835 package are sometimes referred to in the peripherals datasheet as "pads" - a semiconductor design term meaning "chip connection to outside world".
The pads are configurable CMOS push-pull output drivers/input buffers. Register-based control settings are available for

- Internal pull-up / pull-down enable/disable
- Output drive strength
- Input Schmitt-trigger filtering

POWER-ON STATES

All GPIOs revert to general-purpose inputs on power-on reset. The default pull states are also applied, which are detailed in the alternate function table in the ARM peripherals datasheet. Most GPIOs have a default pull applied.

INTERRUPTS

Each GPIO pin, when configured as a general-purpose input, can be configured as an interrupt source to the ARM. Several interrupt generation sources are configurable:

- Level-sensitive (high/low)
- Rising/falling edge
- Asynchronous rising/falling edge

Level interrupts maintain the interrupt status until the level has been cleared by system software (e.g. by servicing the attached peripheral generating the interrupt).

The normal rising/falling edge detection has a small amount of synchronisation built into the detection. At the system clock frequency, the pin is sampled with the criteria for generation of an interrupt being a stable transition within a 3-cycle window, i.e. a record of "1 0 0" or "0 1 1". Asynchronous detection bypasses this synchronisation to enable the detection of very narrow events.

ALTERNATIVE FUNCTIONS

Almost all of the GPIO pins have alternative functions. Peripheral blocks internal to BCM2835 can be selected to appear on one or more of a set of GPIO pins, for example the I2C busses can be configured to at least 3 separate locations. Pad control, such as drive strength or Schmitt filtering, still applies when the pin is configured as an alternate function.

For more detailed information see the Low level peripherals page on the elinuxwiki. There are 54 general-purpose I/O (GPIO) lines split into two banks. All GPIO pins have atleast two alternative functions within BCM. The alternate functions are usually peripheral IOand a single peripheral may appear in each bank to allow flexibility on the choice of IOvoltage. Details of alternative functions are given in section 6.2. Alternative FunctionAssignments.
The block diagram for an individual GPIO pin is given below :

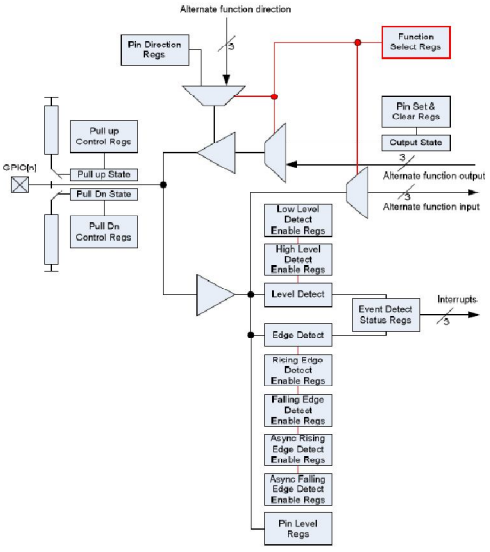


Fig: GPIO block diagram

The GPIO peripheral has three dedicated interrupt lines. These lines are triggered by the setting of bits in the event detect status register. Each bank has its'

own interrupt line with the third line shared between all bits.
The Alternate function table also has the pull state (pull-up/pull-down) which is applied after a power down.

UART

The BCM2835 device has two UARTS. On mini UART and PL011 UART. This section describes the PL011 UART. For details of the mini UART see 2.2 Mini UART.

The PL011 UART is a Universal Asynchronous Receiver/Transmitter. This is the ARM UART (PL011) implementation. The UART performs serial-to-parallel conversion on data characters received from an external peripheral device or modem, and parallel-to-serialconversion on data characters received from the Advanced Peripheral Bus (APB).
The ARM PL011 UART has some optional functionality which can be included or left out.
The UART clock source and associated dividers are controlled by the Clock Manager.
For the in-depth UART overview, please, refer to the ARM PrimeCell UART (PL011)

Troubleshooting UART Problems

The above code works (we've used it for TX and RX). If you can't get to to work for you and you've been through the steps to release the UART from being used for the console try the following:

Baud Rate Error

Try using a slower BAUD rate (or a single 0xFF byte which only has the start bit low) and see if it works. We had a problem using 115k2 baud rate where our microcontroller communicating with the RPi could hit 113636baud or 119047baud. 113636baud had the lowest error margin so we used it and TX from the RPi being received by the microcontroller worked fine. However when transmitting to the RPi nothing was ever received. Changing the microcontroller to use 119047baud caused RX to work. We then tested the RPI transmitting a byte of 0×00 and measured the low state on a scope we got 78uS, showing an actual baud rate of 115384 from the RPi (8bits + the start bit all low). This was odd as 113636baud still had to lower error margin but that was the finding.

Enabling the UART port on a Raspberry PI

By default the serial port on the Raspberry PI is configured for console input/output. If you wish to use the serial port in your own software then you must make the following changes.

Universal SPI Master (2x)

OVERVIEW

The Raspberry Pi is equipped with one SPI bus that has 2 chip selects.
The SPI master driver is disabled by default on Raspian. To enable it, remove the blacklisting for spi-bcm2708 in/etc/modprobe.d/raspi-blacklist.conf, or use raspi-config. Reboot or load the driver manually with:
$ sudomodprobe spi-bcm2708
The SPI bus is available on the P1 Header:
MOSI P1-19
MISO P1-21
SCLK P1-23 P1-24 CE0
GND P1-25 P1-26 CE1
SOFTWARE
WIRINGPIWiringPi includes a library which can make it easier to use the Raspberry Pi's on-board SPI interface. Accesses the hardware registers directly.
http://wiringpi.com/

BCM2835 LIBRARY

This is a C library for Raspberry Pi (RPi). It provides access to GPIO and other IO functions on the Broadcom BCM 2835 chip. Accesses the hardware registers directly.
http://www.airspayce.com/mikem/bcm2835/

USE SPIDEV FROM C

There's a loopback test program in the Linux documentation that can be used as a starting point. See the Troubleshooting section. Uses the Linux spidev driver to access the bus.

SHELL

Write binary 1, 2 and 3
echo-ne "\x01\x02\x03">/dev/spidev0.0

HARDWARE

The BCM2835 on the Raspberry Pi has 3 SPI Controllers. Only the SPI0 controller is available on the header. Chapter 10 in the BCM2835 ARM Peripherals datasheet describes this controller.

MASTER MODES

Signal name abbreviations
SCLK - Serial CLocK
CE - Chip Enable (often called Chip Select)
MOSI - Master Out Slave In
MISO - Master In Slave Out
MOMI - Master Out Master In
MIMO - Master In Master Out

STANDARD MODE

In Standard SPI master mode the peripheral implements the standard 3 wire serial protocol (SCLK, MOSI and MISO).

BIDIRECTIONAL MODE

In bidirectional SPI master mode the same SPI standard is implemented except that a single wire is used for data (MIMO) instead of two as in standard mode (MISO and MOSI).

LOSSI MODE (LOW SPEED SERIAL INTERFACE)

The LoSSI standard allows issuing of commands to peripherals (LCD) and to transfer data to and from them. LoSSI commands and parameters are 8 bits long, but an extra bit is used to indicate whether the byte is a command or parameter/data. This extra bit is set high for a data and low for a command. The resulting 9-bit value is serialized to the output. LoSSI is commonly used with MIPI DBI type C compatible LCD controllers.

Note:

Some commands trigger an automatic read by the SPI controller, so this mode can't be used as a multipurpose 9-bit SPI.

TRANSFER MODES

- Polled
- Interrupt
- DMA

SPEED

The CDIV (Clock Divider) field of the CLK register sets the SPI clock speed:
SCLK = Core Clock / CDIV
If CDIV is set to 0, the divisor is 65536. The divisor must be a power of 2. Odd numbers rounded down. The maximum SPI clock rate is of the APB clock.
Errata: "must be a power of 2" probably should be "must be a multiple of 2"
See the Linux driver section for more info.

CHIP SELECT

Setup and Hold times related to the automatic assertion and de-assertion of the CS lines when operating in **DMA** mode are as follows:

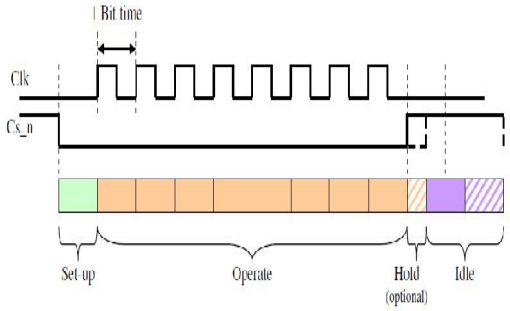
- The CS line will be asserted at least 3 core clock cycles before the msb of the first byte of the transfer.
- The CS line will be de-asserted no earlier than 1 core clock cycle after the trailing edge of the final clock pulse.

LINUX DRIVER

The default Linux driver is spi-bcm2708.
The following information was valid 2014-07-05.

SPI implementation details

The following diagrams shows a typical SPI access cycle. In this case we have 8 SPI clocks.



One bit time before any clock edge changes the CS_n will go low. This makes sure that theMOSI signal has a full bit-time of set-up against any changing clock edges.The operation normally ends after the last clock cycle. Note that at the end there is one halfbittime where the clock does not change but which still is part of the operation cycle.

There is an option to add a half bit cycle hold time. This makes sure that any MISO data hasat least a full SPI bit time to arrive. (Without this hold time, data clocked out of the SPIdevice on the last clock edge would have only half a bit time to arrive).

PCM / I2S Audio

The PCM audio interface is an APB peripheral providing input and output of telephony or high quality serial audio streams. It supports many classic PCM formats including I2S.

The PCM audio interface has 4 interface signals;

PCM_CLK - bit clock.
PCM_FS - frame sync signal.
PCM_DIN - serial data input.
PCM_DOUT - serial data output.

PCM is a serial format with a single bit data_in and single bit data_out. Data is always
serialised MS-bit first.

The frame sync signal (PCM_FS) is used to delimit the serial data into individual frames. The length of the frame and the size and position of the frame sync are fully programmable.

Frames can contain 1 or 2 audio/data channels in each direction. Each channel can be between 8 and 32 bits wide and can be positioned anywhere within the frame as long as the two channels don't overlap. The channel format is separately programmable for transmit and receive directions.

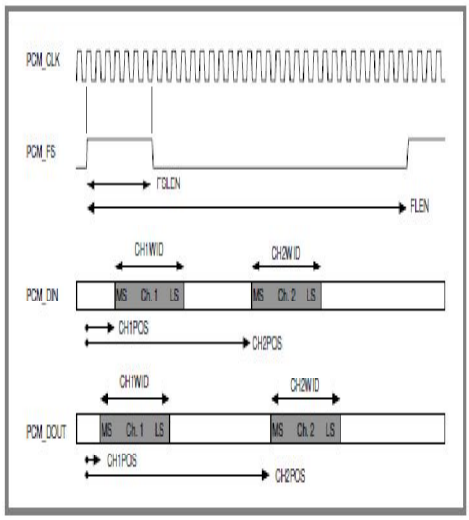


Fig: PCM Audio Interface Typical Timing

The PCM_CLK can be asynchronous to the bus APB clock and can be logically inverted if required.The direction of the PCM_CLK and PCM_FS signals can be individually selected, allowing the interface to act as a master or slave device.The input interface is also capable of supporting up to 2 PDM microphones, as an alternative to the classic PCM input format, in conjunction with a PCM output.

The PCM audio interface contains separate transmit and receive FIFOs. Note that if the frame contains two data channels, they must share the same FIFO and so the channel data will beinterleaved. The block can be driven using simple polling, an interrupt based method or directDMA control.

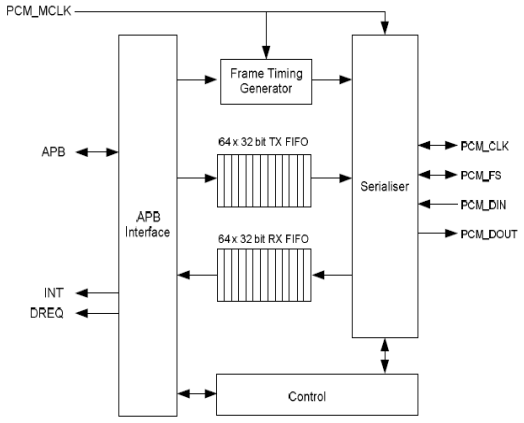


Fig: PCM Audio Interface Block Diagram

Operation:

The PCM interface runs asynchronously at the PCM_CLK rate and automatically transfers transmit and receive data across to the internal APB clock domain. The control registers are NOT synchronised and should be programmed before the device is enabled and should NOT be changed whilst the interface is running.

Only the EN, RXON and TXON bits of the PCMCS register are synchronised across the PCM - APB clock domain and are allowed to be changed whilst the interface is running.The EN bit is a global power-saving enable. The TXON and RXON bits enable transmit and receive, and the interface is running whenever either TXON or RXON is enabled.

In operation, the PCM format is programmed by setting the appropriate frame length, frame sync, channel position values, and signal polarity controls. The transmit FIFO should be preloaded with data and the interface can then be enabled and started, and will run continuously until stopped. If the transmit FIFO becomes empty or the receive FIFO becomes full, the RXERR or TXERR error flags will be set, but the interface will just continue. If the RX FIFO overflows, new samples are discarded and if the TX FIFO underflows, zeros are transmitted.Normally channel data is read or written into the appropriate FIFO as a single word. If the channel is less than 32 bits, the data is right justified and should be padded with zeros. If theRXSEX bit is set then the received data is sign extended up to the full 32 bits. When a frame is programmed to have two data channels, then each channel is written/read as a separateword in the FIFO, producing an interleaved data stream. When initializing the interface, thefirst word read out of the TX FIFO will be used for the first channel, and the data from thefirst channel on the first frame to be received will be the first word written into the RX FIFO.

If a FIFO error occurs in a two channel frame, then channel synchronization may be lost which may result in a left right audio channel swap. RXSYNC and TXSYNC status bits areprovided to help determine if channel slip has occurred. They indicate if the number of words in the FIFO is a multiple of a full frame (taking into account where we are in the current frame being transferred). This assumes that an integer number of frames data has beensent/read from the FIFOs.

Note that data is always serialized MS-bit first. This is well-established behavior in both PCM and I2S.If the PDM input mode is enabled then channel 1 is sampled on the negative edge of PCM_CLK whilst channel 2 is sampled on the positive edge of PCM_CLK.

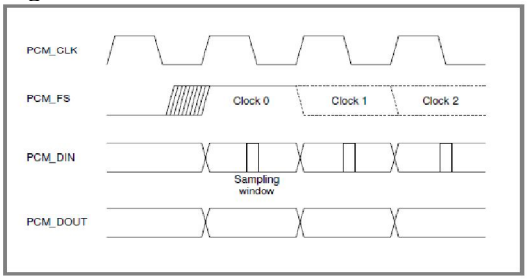


Fig: Timing at Start of Frame

Note that the precise timing of FS (when it is an input) is not clearly defined and it may change state before or after the

positive edge of the clock. Here the first clock of the frame is defined as the clock period where the PCM_FS is sampled (negative edge) as a 1 where itwas previously sampled as a 0.

POWER SUPPLY

The device is powered by a 5V micro USB supply. Exactly how much current (mA) the Raspberry Pi requires is dependent on what you connect to it. We have found that purchasing a 1.2A (1200mA) power supply from a reputable retailer will provide you with ample power to run your Raspberry Pi.

Typically, the model B uses between 700-1000mA depending on what peripherals are connected; the model A can use as little as 500mA with no peripherals attached. The maximum power the Raspberry Pi can use is 1 Amp. If you need to connect a USB device that will take the power requirements above 1 Amp, then you must connect it to an externally-powered USB hub.

BACKPOWERING

Backpowering occurs when USB hubs do not provide a diode to stop the hub from powering against the host computer. Other hubs will provide as much power as you want out each port.

USB

OVERVIEW

The Raspberry Pi Model B is equipped with two USB2.0 ports. These are connected to the LAN9512 combo hub/Ethernet chip IC3, which is itself a USB device connected to the single upstream USB port on BCM2835.

On the Model A, the single USB2.0 port is directly wired to BCM2835.

SUPPORTED DEVICES

In general, every device supported by Linux is possible to use with the Pi, subject to a few caveats detailed further down. Linux has probably the most comprehensive driver database for legacy hardware of any operating system (it can lag behind for modern device support as it requires open-source drivers for Linux to recognize the device by default).

If you have a device and wish to use it with a Pi, then plug it in. Chances are that it'll "just work". If you are running in a graphical interface (such as the LXDE desktop environment in Raspbian), then it's likely that an icon or similar will pop up announcing the new device.

If the device doesn't appear to work, then refer to the Troubleshooting section.

GENERAL LIMITATIONS

The OTG hardware on Raspberry Pi has a simpler level of support for certain devices, which may present a higher software processing overhead. The Raspberry Pi also has only one root USB port: all traffic from all connected devices is funneled down this bus, which operates at a maximum speed of 480mbps.

The USB specification defines three device speeds - Low, Full and High. Most mice and keyboards are Low-speed, most USB sound devices are Full-speed and most video devices (webcams or video capture) are High-speed.

Software Sources

APT keeps a list of software sources on your Pi in a file at /etc/apt/sources.list. Before installing software, you should update your package list with apt-get update:

sudo apt-get update

Updating and Upgrading Raspbian

There are two steps to upgrading. First, run sudo apt-get update in order to synchronize the database of available software packages and the versions available. Next, run sudo apt-get upgrade which will cause any packages with newer versions available to be updated.

Generally speaking, doing this regularly will keep your installation up to date; in other words, it will be equivalent to the latest released image. However, there are occasionally times where a change is

made in the Foundation's Raspbianimage that require your intervention to reproduce. A recent example is the addition of the xserver-xorg-video-fbturbo X.Org driver to the standard image; this requires users running older images to manually install the package to benefit from it. Cases like this are documented on the relevant image update announcement on the Raspberry Pi blog.

UPDATING THE KERNEL AND FIRMWARE

The kernel and firmware are installed as a Debian package, and so will also get updates when using the procedure above. These packages are updated infrequently (after extensive testing); if you want to try more recent experimental software, it's also easy to update to the latest available version using rpi-update. This is pre-installed on the current Raspbian image, so you can just use sudorpi-update to try the latest firmware; this will sometimes be suggested when troubleshooting. If you receive errors about invalid certificates, then run sudo apt-get update &&sudo apt-get install rpi-update to upgrade to the latest rpi-update version.

Running Out Of Space

When running sudo apt-get upgrade, it will show how much data will be downloaded and how much space it will take up on the SD card. It's worth checking with df -h that you have enough disk space free, as unfortunately apt will not do this for you. Also be aware that downloaded package files (.deb files) are kept in /var/cache/apt/archives. You can remove these in order to free up space with sudo apt-get clean.

RASPI-CONFIG

raspi-config is the Raspberry Pi configuration tool written and maintained by Alex Bradbury. It targets Raspbian.

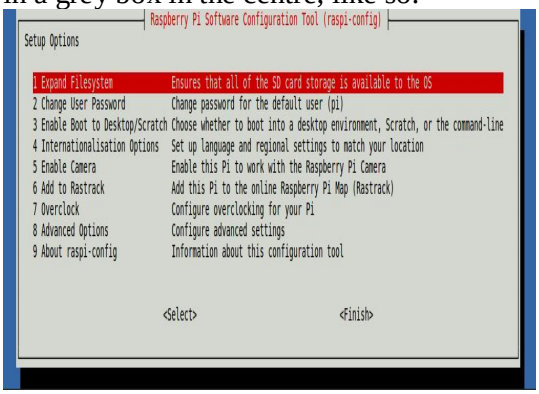
USAGE

You will be shown raspi-config on first booting into Raspbian. To open the configuration tool after this, simply run the following from the command line:

sudoraspi-config

The sudo is required because you will be changing files that you do not own as the pi user.

You should see a blue screen with options in a grey box in the centre, like so:



It has the following options available:

Raspberry Pi Software Configuration Tool (raspi-config)

Moving Around the Menu

Use the up and down arrow keys to move the highlighted selection between the options available. Pressing the right arrow key will jump out of the options menu and take you to the <Select> and <Finish> buttons. Pressing left will take you back to the options. Alternatively, use the Tabkey to switch between these.

Note that in long lists of option values (like the list of timezone cities), you can also type a letter to skip to that section of the list. For example, enteringL will skip you to Lisbon, just two options away from London, to save you scrolling all the way through the alphabet.

What Raspi-Config Does

Generally speaking, raspi-config aims to provide the functionality to make the most common configuration changes. This may result in automated edits to /boot/config.txt and various standard Linux configuration files. Some options require a reboot to take effect. If you

changed any of those, raspi-config will ask if you wish to reboot now when you select the <Finish>button.

SUDO

You will not normally log into to the computer as root, but can instead use the sudo command to provide access as the superuser. If you log into your Raspberry Pi as the pi user then you are logging in as a normal user. You can run commands as the root user by using the sudo command before the program you want to run.

For example if you want to install additional software on Raspbian then you normally use the apt-get tool. To be able to update the list of available software then you need to prefix the apt-get command command with sudo. sudo apt-get update

Find out more about the apt commands

You can also run a superuser shell by using sudosu. When running commands as a superuser there is nothing to protect against mistakes that could damage the system. It is like disabling the safety guards on a machine. It makes it easier to access the insides, but the risk of damage is far greater. It is recommended that you only run commands as the superuser when required and to exit a superuser shell when it is no longer needed.

Who Can Use Sudo?

It would defeat the point of the security if anyone could just put sudo in front of their commands, so only approved users can use sudo to gain administrator privileges. The pi user is included in the sudoers file. To allow other users to act as a superuser then you could add the user to the sudo group or add them using visudo.

Pi Store

The Pi Store is the Raspberry Pi app store - you can browse and download community developed software and games, and even submit your own applications for others to download for free or to buy.

You can browse the Pi Store online at store.raspberrypi.com

Read Eben's announcement of the Pi Store here: Introducing the Pi Store

Kernel

The Raspberry Pi kernel is stored in GitHub and can be viewed atgithub.com/raspberrypi/linux; it follows behind the main linux kernel.

The main Linux kernel is continuously updating; we take long-term releases of the kernel, which are mentioned on the front page, and integrate the changes into the Raspberry Pi kernel. We then create a 'next' branch which contains an unstable port of the kernel; after extensive testing and discussion we push this to the main branch.

- Updating your kernel
- Building a new kernel
- Configuring the kernel
- Applying patches to the kernel
- Getting the kernel headers

Getting Your Code Into The Kernel

There are many reasons you may want to put something into the kernel:

- You've written some Raspberry Pi-specific code that you want everyone to benefit from
- You've written a generic Linux kernel driver for a device and want everyone to use it
- You've fixed a generic kernel bug
- You've fixed a Raspberry Pi-specific kernel bug

Initially you should fork the Linux repository and clone that on your build system; this can be either on the Raspberry Pi or on a Linux machine you're cross-compiling on. You can then make your changes, test them, and commit them into your fork.

Next, depending upon whether the code is Raspberry Pi-specific or not:

For Pi-specific changes or bug fixes, submit a pull request to the kernel.

For general Linux kernel changes (i.e. a new driver) these need to be submitted upstream first. Once they've been submitted upstream

and accepted, submit the pull request and we'll receive it.

Building Qt 5 on Raspberry Pi

This is a HOW TO guide for building Qt 5 for the Raspberry Pi, and building and deploying Qt 5 apps using Qt Creator. This guide will be using Raspbian "Wheezy", a Debian based distro designed for the Raspberry Pi. This guide also assumes the use of Linux or UNIX on the workstation side.

Install a Toolchain

To build on the Raspberry Pi we need a cross-compile toolchain. The toolchain will contain compilers, linkers and other tools that run on the host workstation but create executables for the target Raspberry Pi.

Ethernet Module

This board utilizes the new Microchip ENC28J60 Stand- Alone Ethernet Controller IC featuring a host of features to handle most of the network protocol requirements. The board connects directly to most microcontrollers with a standard SPI interface with a transfer speed of up to 20 MHz. This module contains what you need to enable ethernet on your next microcontroller project. It includes a ENC28J60 ethernet controller, an RJ45 socket with link/activity lights and integrated transformer. This ethernet module enables you to connect a particular embedded device (equipped with SPI support) on to a network. By using this Ethernet module (along with a microcontroller running a small TCP/IP stack) applications like Embedded Web server can be easily developed.

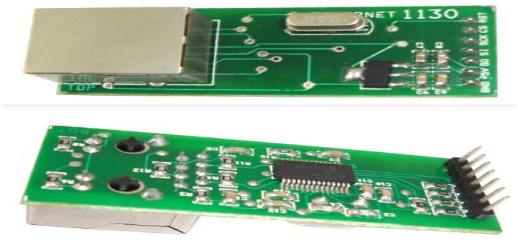


Fig: Ethernet Module

Use this module to enable Ethernet interface for your product. It works with any microcontroller operating at 3.3V or 5V. This module works at 5V and is compatible with 5V interface lines. Use SPI process to interface with it. Host web server, ping the module or add it to home automation via internet. Heart of this module is ENC28J60 Ethernet controller from Microchip. This board is assembled with all components as shown in the picture.

The circuit board includes all required components for the ethernet controller, plus a 3.3V Voltage Regulator and a RJ-45 jack with integrated transformer and built-in Link and Activity LEDs for connection to an Ethernet Local Area Network.

Precision Centigrade Temperature Sensor

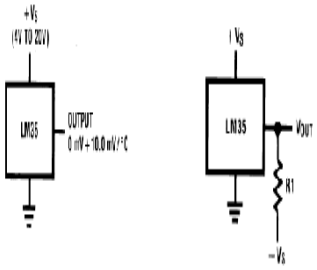
LM35 converts temperature value into electrical signals. LM35 series sensors are precision integrated-circuit temperature sensors whose output voltage is linearly proportional to the Celsius temperature. The LM35 requires no external calibration since it is internally calibrated. . The LM35 does not require any external calibration or trimming to provide typical accuracies of ±1⁄4°C at room temperature and ±3⁄4°C over a full −55 to +150°C temperature range.

The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air.

Features

1. Calibrated directly in ° Celsius (Centigrade)

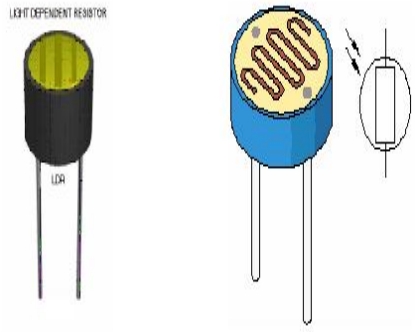
2. Linear + 10.0 mV/°C scale factor
3. 0.5°C accuracy guaranteed (at +25°C)
4. Rated for full −55° to +150°C range
5. Suitable for remote applications
6. Low cost due to wafer-level trimming
7. Operates from 4 to 30 volts
8. Less than 60 µA current drain
9. Low self-heating, 0.08°C in still air
10. Nonlinearity only ±1⁄4°C typical
11. Low impedance output, 0.1 W for 1 mA load



The characteristic of this LM35 sensor is:
For each degree of centigrade temperature it outputs 10milli volts.

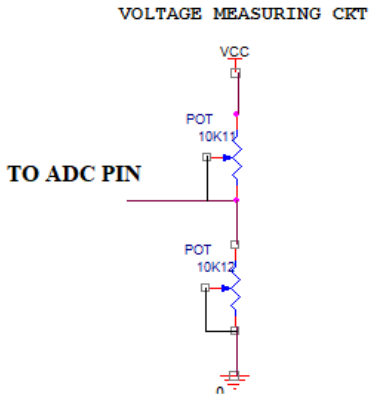
Light Dependent Resistor:

LDRs or Light Dependent Resistors are very useful especially in light/dark sensor circuits. Normally the resistance of an LDR is very high, sometimes as high as 1,000,000 ohms, but when they are illuminated with light, the resistance drops dramatically.Thus in this project, LDR plays an important role in switching on the lights based on the intensity of light i.e., if the intensity of light is more (during daytime) the lights will be in off condition. And if the intensity of light is less (during nights), the lights will be switched on.



The output of the LDR is given to ADC which converts the analog intensity value into corresponding digital data and presents this data as the input to the microcontroller.

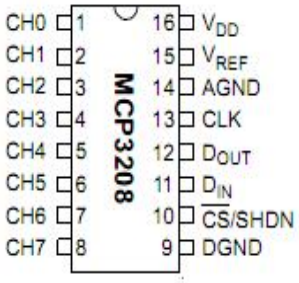
Voltage measuring circuit:



The supply to the voltage measuring circuit is the Adaptor. Thus, the voltage coming from the adaptor is applied to two potentiometers and the middle point of the two potentiometers is given to ADC pin of the microcontroller.Thus the voltage value will be displayed on LCD.

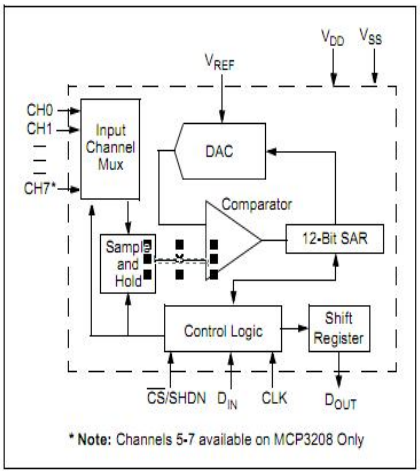
ADC- MCP3208

The MCP3208 12-bit Analog-to-Digital Converter (ADC) combines high performance and low power consumption in a small package, making it ideal for embedded control applications. The MCP3208 features a successive approximation register (SAR) architecture and an industry-standard SPI™ serial interface, allowing 12-bit ADC capability to be added to any processor.



Functional block diagram:

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 06 Issue 13
December 2019



The MCP3208 features 100k /second, 8 input channels, low power consumption (5nA typical standby, 400 µA max.active), and is available in 16-pin PDIP and SOIC packages. Applications for the MCP3208 include data acquisition, instrumentation and measurement, multi-channel data loggers, industrial PCs, motor control, robotics, industrial automation, smart sensors, portable instrumentation and home medical appliances.

Pin Function Table

Name	Function
V_{DD}	+2.7V to 5.5V Power Supply
DGND	Digital Ground
AGND	Analog Ground
CH0-CH7	Analog Inputs
CLK	Serial Clock
D_{IN}	Serial Data In
D_{OUT}	Serial Data Out
$\overline{CS}$/SHDN	Chip Select/Shutdown Input
V_{REF}	Reference Voltage Input

Applications

- Sensor Interface
- Process Control
- Data Acquisition
- Battery Operated Systems

PIN DESCRIPTIONS

CH0-CH7

These are analog inputs for channels 0 to 7 respectively. These channels can programmed to be used as two independent channels in single ended-mode or as a single pseudo-differential input where one channel is IN+ and one channel is IN-.

Chip Select/Shutdown (CS/SHDN)

The CS/SHDN pin is used to initiate communication with the device when pulled low and will end a conversion and put the device in low power standby when pulled high. The CS/SHDN pin must be pulled high between conversions.

Serial Clock (CLK)

The SPI clock pin is used to initiate a conversion and to clock out each bit of the conversion as it takes place.

Serial Data Input (DIN)

The SPI port serial data input pin is used to clock in input channel configuration data.

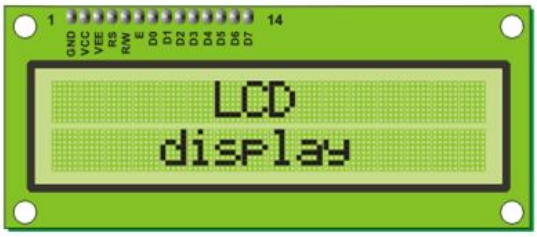
Serial Data Output (DOUT)

The SPI serial data output pin is used to shift out the results of the A/D conversion. Data will always change on the falling edge of each clock as the conversion takes place.

Liquid Crystal Display:

LCD stands for Liquid Crystal Display. LCD is finding wide spread use replacing LEDs (seven segment LEDs or other multi segment LEDs) because of the following reasons:

1. The declining prices of LCDs.
2. The ability to display numbers, characters and graphics. This is in contrast to LEDs, which are limited to numbers and a few characters.
3. Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD. In contrast, the LED must be refreshed by the CPU to keep displaying the data.
4. Ease of programming for characters and graphics.



These components are "specialized" for being used with the microcontrollers, which means that they cannot be activated by standard IC circuits. They are used for writing different messages on a miniature LCD.

A model described here is for its low price and great possibilities most frequently used in practice. It is based on the HD44780 microcontroller (*Hitachi*) and can display messages in two lines with 16 characters each. It displays all the alphabets, Greek letters, punctuation marks, mathematical symbols etc. In addition, it is possible to display symbols that user makes up on its own. Automatic shifting message on display (shift left and right), appearance of the p ointer, backlight etc. are considered as useful characteristics.

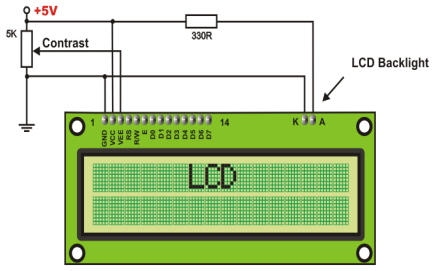
Pins Functions

There are pins along one side of the small printed board used for connection to the microcontroller. There are total of 14 pins marked with numbers (16 in case the background light is built in). Their function is described in the table below:

Function	Pin Number	Name		Description
Ground	1	Vss	-	0V
Power supply	2	Vdd	-	+5V
Contrast	3	Vee	-	0 - Vdd
Control of operating	4	RS	0 1	D0 – D7 are interpreted as commands D0 – D7 are interpreted as data
	5	R/W	0 1	Write data (from controller to LCD) Read data (from LCD to controller)
	6	E	0 1 From 1 to 0	Access to LCD disabled operating Data/commands are transferred to LCD
Data / commands	7	D0	0/1	Bit 0 LSB
	8	D1	0/1	Bit 1
	9	D2	0/1	Bit 2
	10	D3	0/1	Bit 3
	11	D4	0/1	Bit 4
	12	D5	0/1	Bit 5
	13	D6	0/1	Bit 6
	14	D7	0/1	Bit 7 MSB

LCD screen:

LCD screen consists of two lines with 16 characters each. Each character consists of 5x7 dot matrix. Contrast on display depends on the power supply voltage and whether messages are displayed in one or two lines. For that reason, variable voltage 0-Vdd is applied on pin marked as Vee. Trimmer potentiometer is usually used for that purpose. Some versions of displays have built in backlight (blue or green diodes). When used during operating, a resistor for current limitation should be used (like with any LE diode).



LCD Basic Commands

All data transferred to LCD through outputs D0-D7 will be interpreted as commands or as data, which depends on logic state on pin RS:

- RS = 1 - Bits D0 - D7 are addresses of characters that should be displayed. Built in processor addresses built in "map of characters" and displays corresponding symbols. Displaying position is determined by DDRAM address. This

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 06 Issue 13
December 2019

address is either previously defined or the address of previously transferred character is automatically incremented.

- RS = 0 - Bits D0 - D7 are commands which determine display mode. List of commands which LCD recognizes are given in the table below:

Command	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Execution Time
Clear display	0	0	0	0	0	0	0	0	0	1	1.64mS
Cursor home	0	0	0	0	0	0	0	0	1	x	1.64mS
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	40uS
Display on/off control	0	0	0	0	0	0	1	D	U	B	40uS
Cursor/Display Shift	0	0	0	0	0	1	D/C	R/L	x	x	40uS
Function set	0	0	0	0	1	DL	N	F	x	x	40uS
Set CGRAM address	0	0	0	1	CGRAM address						40uS
Set DDRAM address	0	0	1	DDRAM address							40uS
Read "BUSY" flag (BF)	0	1	BF	DDRAM address							-
Write to CGRAM or DDRAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	40uS
Read from CGRAM or DDRAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	40uS

I/D 1 = Increment (by 1)
0 = Decrement (by 1)
S 1 = Display shift on
0 = Display shift off
D 1 = Display on
0 = Display off
U 1 = Cursor on
0 = Cursor off
B 1 = Cursor blink on
0 = Cursor blink off

R/L 1 = Shift right
0 = Shift left
DL 1 = 8-bit interface
0 = 4-bit interface
N 1 = Display in two lines
0 = Display in one line
F 1 = Character format 5x10 dots
0 = Character format 5x7 dots
D/C 1 = Display shift
0 = Cursor shift

LCD Initialization:

Once the power supply is turned on, LCD is automatically cleared. This process lasts for approximately 15mS. After that, display is ready to operate. The mode of operating is set by default. This means that:

1. Display is cleared
2. Mode

DL = 1 Communication through 8-bit interface
N = 0 Messages are displayed in one line
F = 0 Character font 5 x 8 dots

3. Display/Cursor on/off

D = 0 Display off
U = 0 Cursor off
B = 0 Cursor blink off

4. Character entry

ID = 1 Addresses on display are automatically incremented by 1
S = 0 Display shift off

Automatic reset is mainly performed without any problems. Mainly but not always! If for any reason power supply voltage does not reach full value in the course of 10mS, display will start perform completely unpredictably. If voltage supply unit cannot meet this condition or if it is needed to provide completely safe operating, the process of initialization by which a new reset enabling display to operate normally must be applied.

Algorithm according to the initialization is being performed depends on whether connection to the microcontroller is through 4- or 8-bit interface. All left over to be done after that is to give basic commands and of course- to display messages.

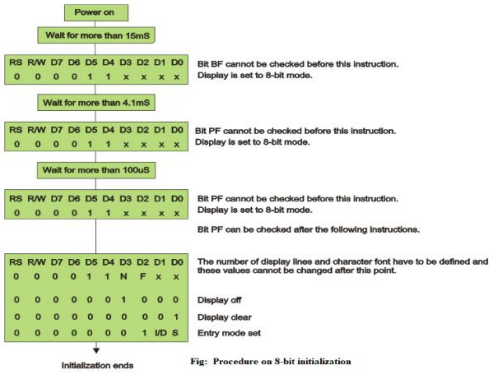


Fig: Procedure on 8-bit initialization

Firmware Implementation Of The Project Design

This chapter briefly explains about the firmware implementation of the project. The required software tools are discussed in the following section.

FIRMWARE IMPLEMENTATION

Raspbian is a competent and versatile operating system that gives your Raspberry Pi all the comforts of a PC: a command line, a browser, and tons of other programs. You can use a Raspberry Pi running Raspbian as a cheap and effective home computer, or

you can use it as a springboard and turn your Raspberry Pi into any of countless other functional devices, from wireless access points to retro gaming machines. Here's how to install Raspbian on the Raspberry Pi.

How to install Raspbian on the Raspberry

PiInstalling Raspbian on the Raspberry Pi is pretty straightforward. We'll be downloading Raspbian and writing the disc image to a microSD card, then booting the Raspberry Pi to that microSD card. For this project, you'll need a microSD card (go with at least 8 GB), a computer with a slot for it, and, of course, a Raspberry Pi and basic peripherals (a mouse, keyboard, screen, and power source). This isn't the only method for installing Raspbian (more on that in a moment), but it's a useful technique to learn because it can also be used to install so many other operating systems on the Raspberry Pi. Once you know how to write a disc image to a microSD card, you open up a lot of options for fun Raspberry Pi projects.

A word about NOOBS

It's worth noting that the method described here isn't your only option for installing Raspbian. You can also opt to use NOOBS, an operating system installation manager that makes it easy to install Raspbian, as well as a few other operating systems.

If you really want to make things easy, you can even buy SD cards that come pre-loaded with NOOBS.

Step 1: Download Raspbian

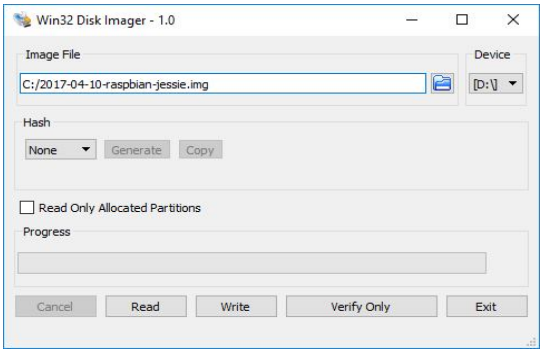
First things first: hop onto your computer (Mac and PC are both fine) and download the Raspbian disc image. **You can find the latest version of Raspbian on the Raspberry Pi Foundation's website here.**

Step 2: Unzip the file

The Raspbian disc image is compressed, so you'll need to unzip it. The file uses the ZIP64 format, so depending on how current your built-in utilities are, you need to use certain programs to unzip it. If you have any trouble, try these programs recommended by the Raspberry Pi Foundation:

- Windows users, you'll want **7-Zip.**
- Mac users: **The Unarchiver** is your best bet.
- Linux users will use the appropriately named **Unzip**.

Step 3: Write the disc image to your microSD card



Next, pop your microSD card into your computer and write the disc image to it. You'll need a specific program to do this:

- Windows users, your answer is **Win32 Disk Imager**.
- Mac users, you can use the disk utility that's already on your machine.
- Linux people, **Etcher**, which also works on Mac and Windows is what the Raspberry Pi Foundation recommends.

The process of actually writing the image will be slightly different across these programs, but it's pretty self-explanatory no matter what you're using. Each of these

programs will have you select the destination (make sure you've picked your microSD card!) and the disc image (the unzipped Raspbian file). Choose, double-check, and then hit the button to write.

Step 4: Put the microSD card in your Pi and boot up

Once the disc image has been written to the microSD card, you're ready to go. Put the card into your Raspberry Pi, plug in the peripherals and power source. The current edition to Raspbian will boot directly to the desktop. Your default credentials are username **pi** and password **raspberry.**

Working procedure:

IOT Based Data acquisition system is developed using powerful development platform Raspberry Pi board. This system uses Temperature sensor (LM35), Light Intensity Sensor (LDR) and voltage measuring circuit. All these sensors are interfaced with GPIO header of Raspberry Pi board.

The main reasons why we have chosen this specific element are the high processing capacity, relatively low price, and its ability to adapt in different programming modes. The device uses Linux as an operating system, which has access to a large number of libraries and applications compatible with it. Raspberry Pi has an Ethernet port allowing us a network connection, as long as we are in the same subnet with the device we want to access and manage, 4 USB ports used to connect devices like a keyboard, mouse, camera, and other devices that connect through a USB port, and an HDMI port giving us access to the interface of the operating system installed, and can also be used the first time while installing the devices.

It has 40 pins that allow us to receive and send signals. They are divided in half into two groups: the 3V, and the 5V group. Therefore, one side of the microcontroller gives a voltage of 3V, and the other 5V. Besides the 40 voltage pins, it has pins that are used to receive signals, which in our case was used to connect the button, that will send the signal for the face identification. Raspberry Pi does not have an operating system previously installed, but that can be downloaded from the Raspberry website, and transferred to an SD card.

The sensors are continuously read by the Raspberry pi board and will be sent to web page by Python programming. This sensor's data will be uploaded to user mobile using the web link IFTTT protocol. The user can view this sensor's parameters on his/her mobile phone from anywhere at any time.

Applications:

In home automation.
Industrial automation and Military and defense applications.
Electrical applications.

Futurescope:

By adding more sensors we automate the places such as industries, military…etc. Can be control sensor parameters with by adding load to controller at particle values. Sending Email alerts to the user when Higher values occur.

References:

[1] Bhargav Garodiya & H. N. Pandya, Real Time Monitoring & Data Logging System using Arm architecture of Raspberry Pi & Arduino UNO, International Journal of VLSI and Embedded Systems-IJVES, Vol 04, Article 06118, 2013.
[2] https://www.raspberrypi.org
[3] K. Anitha, M. Srinivasa Rao & C. R. S. Murthy, "Embedded Bluetooth Data Acquisition System based on ARM for

Unmanned Underwater Vehicle (UUV)", IJCSNS International Journal of Computer Science and Network Security, VOL. 11 No. 7, 2011

[4] M. F. AL. Faisal, S Bakar & PS Rudati, "The Development of A Data Acquisition System Based on Internet of Things Framework", ICT for Smart Society (ICISS), 2014 IEEE, 10.1109/ICTSS.2014.7013175, 2014.

[5] Mirjana Maksimović, Vladimir Vujović, Nikola Davidović, Vladimir Milošević & Branko Perišić, "Raspberry Pi as Internet of Things hardware: Performances and Constraints", ResearchGate, At Vrnjacka Banja, Serbia, IcETRAN, 2014