CHAPTER – 1

INTRODUCTION

1.1 OBJECTIVE

Blindness is the condition of lacking visual observation due to neurological and physiological factors. For blind pedestrian secure mobility is one of the biggest challenges faced in their daily life. According to the World Health Organization (WHO) in 2012, out of 7 billion global population there were over 285 million visually impaired people and 39 million were totally blind out of which 19 million are children (below 15 years) and this number is growing at an alarming rate. So, some navigation system is required to assist or guide this people. Many researches are being conducted to build navigation system for blind people. Most of these technologies have boundaries as its challenge involves accuracy, interoperability, usability, coverage which is not easy to overcome with current technology for both indoor and outdoor navigation. The proposed system mainly focuses on two components; sensing of the immediate surrounding environment against obstacle for the visually impaired person and warning about the obstacle by means of voice feedback system.

1.2 ABSTRACT

According to a recent report of the World Health Organization 81.7% of all 39 million blind people worldwide are 50 years and older. These people have an inherent risk towards walking disabilities**.** In present days, the navigation system to help for blind peoples our daily life. The proposed systems were based on ultrasonic sensor to detect the obstacles and alert the blind person. This information is useful for the blind to navigate themselves. For example, the user can program locations like Home, Office, Bus stop etc on database on the user device. The database is programmed with the location co-ordinates of these places. The location co ordinate of the blind path gives the information about the current location. The user is provided with a customized embedded systems based device, which has a keypad. The user can select the location they want to go to using the keypad. This information is provided as a voice using android application. Thus the user can independently navigate them using just voice commands and listening to the directions provided by the device.

1.3 LITERATURE SURVEY

Title: A Wireless Navigation System For the Visually Impaired

This paper presents an integrated wireless system that helps a visually impaired person navigate within an indoor environment. The guidance process to help the person navigate uses a ZigBee wireless mesh network to localize the user and a compass to determine his/her orientation.

Title: A Blind Navigation System Using RFID for Indoor Environments

We have proposed an RFID system for blind navigation which can be used by blind people, by tourists, or by fire fighters for a rescue in a building with smoke. A system prototype which includes RFID tags embedded in a footpath block, the embedded system as a navigation device, and a navigation server which is remotely connected to the device via the Internet.

Title: Navigation System for Visually Impaired People

Navigation assistance for visually impaired (NA VI) refers to systems that are able to assist or guide people with vision loss, ranging from partially sighted to totally blind, by means of sound commands.

Title: RFID and GPS integrated navigation system for the visually impaired

This paper describes an RFID and GPS integrated navigation system, Smart-Robot (SR) for the visually impaired. The SR uses RFID and GPS based localization while operating indoor and outdoor respectively. The portable terminal unit is an embedded system equipped with an RFID reader, GPS, and analog compass as input devices to obtain location and orientation.

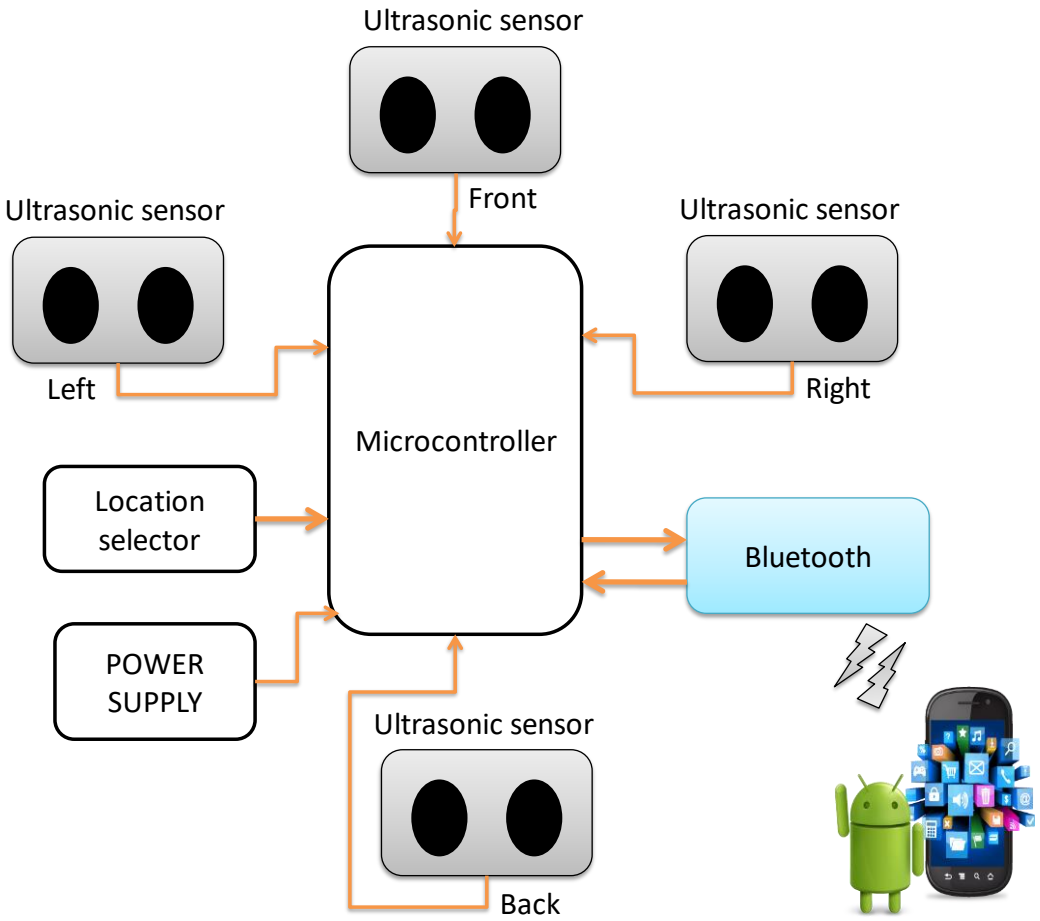
Title: Supporting learning for individuals with visual impairment

This paper presents an embodied perspective on the development of multimodal and perceptual replacement technologies for Individuals with Blindness or Severe Visual Impairment (IBSVI)

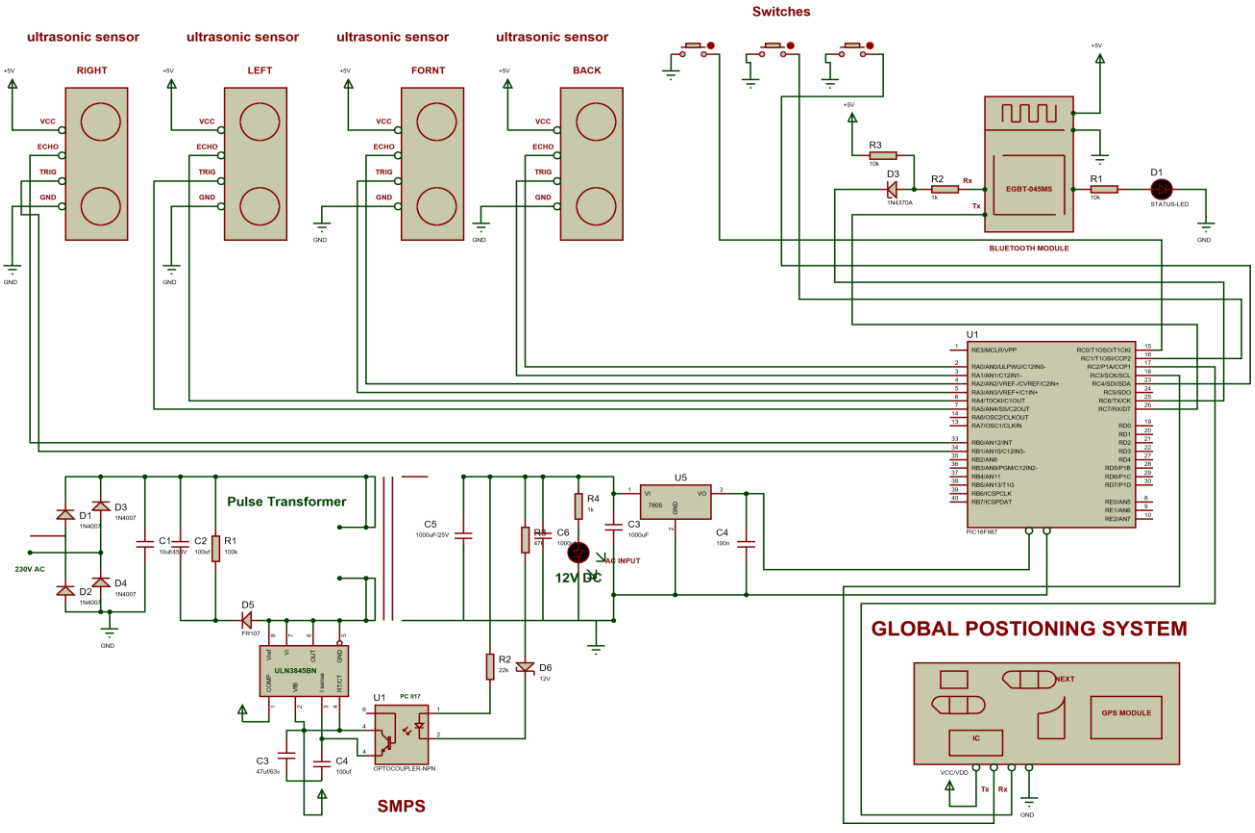
CHAPTER- 2

BLOCK DIAGRAM

2.1 BLOCK DIAGRAM



CIRCUIT DIAGRAM


Switches
ultrasonic sensor
ultrasonic sensor
ultrasonic sensor
ultrasonic sensor
RIGHT
LEFT
FORNT
BACK
VCC
ECHO
TRIG
GND
R3
D3
R2
EGBT-045MS
BLUETOOTH MODULE
R1
D1
STATUS-LED
U1
PIC16F887
Pulse Transformer
230V AC
D1
D2
D3
D4
C1
C2
R1
D5
C5
R4
C6
C3
U5
C4
12V DC
AC INPUT
ULN3845BN
U1
PC 817
OPTOCOUPLER-NPN
R2
D6
C3
C4
SMPS
GLOBAL POSTIONING SYSTEM
NEXT
GPS MODULE
IC
VCC/VDD
Tx
Rx
GND

2.3 CIRCUIT DESCRIPTION

Our circuit diagram consists of Ultrasonic sensors, power supply section, Bluetooth and GPS. Firstly 230V AC will be converted in to 9V AC by transformer. Then that will be rectified using bridge rectifier circuit. finally filtered DC output will be given to regulator 7805. that will be given as power supply to sensors, Bluetooth and controller. we are using PIC16F887 microcontroller to control the entire system. it comprises of 40 pins as 5 ports and 14 ADC pins. ultrasonic sensor comprises of 4 pins. Namely, vcc, grd, echo, trigger. echo and trigger pins will be connected to controller analog pins. we are using totally four ultrasonic sensors for left, right, front, back side objects detection. so ultrasonic sensors will use totally 8 analog pins. Then location selector switches are used to select location to which blind person want to move. This will be connected to digital pins of controller. Bluetooth Tx, Rx pins will be connected with Rx and Tx pins of controller in order to transmit the data to android app. GPS will be used to sense the location of blind person.

CHAPTER - 3

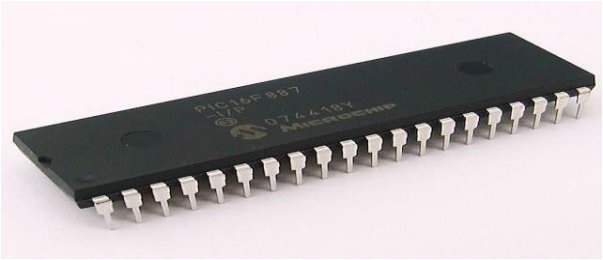
HARDWARE

3.1 MICROCONTROLLER

- A single chip that contains
  - Processor (the CPU)
  - Non-volatile memory for the program (ROM or flash)
  - Volatile memory for input and output (RAM)
  - Clock
  - An I/O control unit
- Also called a “computer on a chip”
- Billions of microcontroller units (MCUs) are embedded each year in a large number of products from toys to automobiles.
- For example: a single vehicle can use 70 or more microcontrollers.
- Microcontrollers are designed for embedded applications, in contrast to the microprocessor used in personal computers or other general purpose applications.
- PIC microcontroller chips from Microchip are the world’s smallest microcontrollers.

PIC MICROCONTROLLER

Peripheral Interface Controller (PIC) was originally designed by General Instruments. In the late 1970s, GI introduced PIC 1650 and 1655 – RISC with 30 instructions. PIC was sold to Microchip Features: low-cost, self-contained, 8-bit, Harvard structure, pipelined, RISC, single accumulator, with fixed reset and interrupt vectors.



REASON FOR USING PIC

- Variety of choices (8-bit to 32-bit)
- Affordable (Low Cost)
- Low Power
- Reasonable Size
- Convenient Packaging
- Through Hole (Dip)
- Surface Mount (SMD)

MICROCONTROLLER FEATURES

- High-Performance RISC CPU.
- Only 35 instructions to learn.
- Operating speed.
- Interrupt capability.
- 8-level deep hardware stack.
- Direct, Indirect and Relative Addressing modes.
- Special Microcontroller Features.
- Precision Internal Oscillator.
- Power-Saving Sleep mode.
- Wide operating voltage range (2.0V-5.5V).
- Industrial and Extended Temperature range.
- Power-on Reset (POR).
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST) .
- Brown-out Reset (BOR) with software control option.
- Enhanced low-current Watchdog Timer (WDT) with on-chip oscillator (software selectable nominal 268 seconds with full prescaler) with software enable.
- Multiplexed Master Clear with pull-up/input pin.
- Programmable code protection.
- High Endurance Flash/EEPROM cell.

- Program memory Read/Write during run time.
- In-Circuit Debugger (on board).
- Low-Power Features.
- Standby Current.
- Operating Current.
- Watchdog Timer Current.
- Peripheral Features.
- 24/35 I/O pins with individual direction control.
- Analog Comparator module with.
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler.
- Enhanced Timer1.
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler.
- Supports RS-485, RS-232, and LIN 2.0
- Auto-Baud Detect
- Auto-Wake-Up on Start bit
- In-Circuit Serial Programming.
- Enhanced Capture, Compare, PWM+ module.
- Capture, Compare, PWM module.
- Enhanced USART module.
- In-Circuit Serial Programming TM (ICSPTM) via two pins.
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI (all 4 modes) and I2C™ Master and Slave Modes with I2C address mask.

PIN DIAGRAM OF PIC16F887

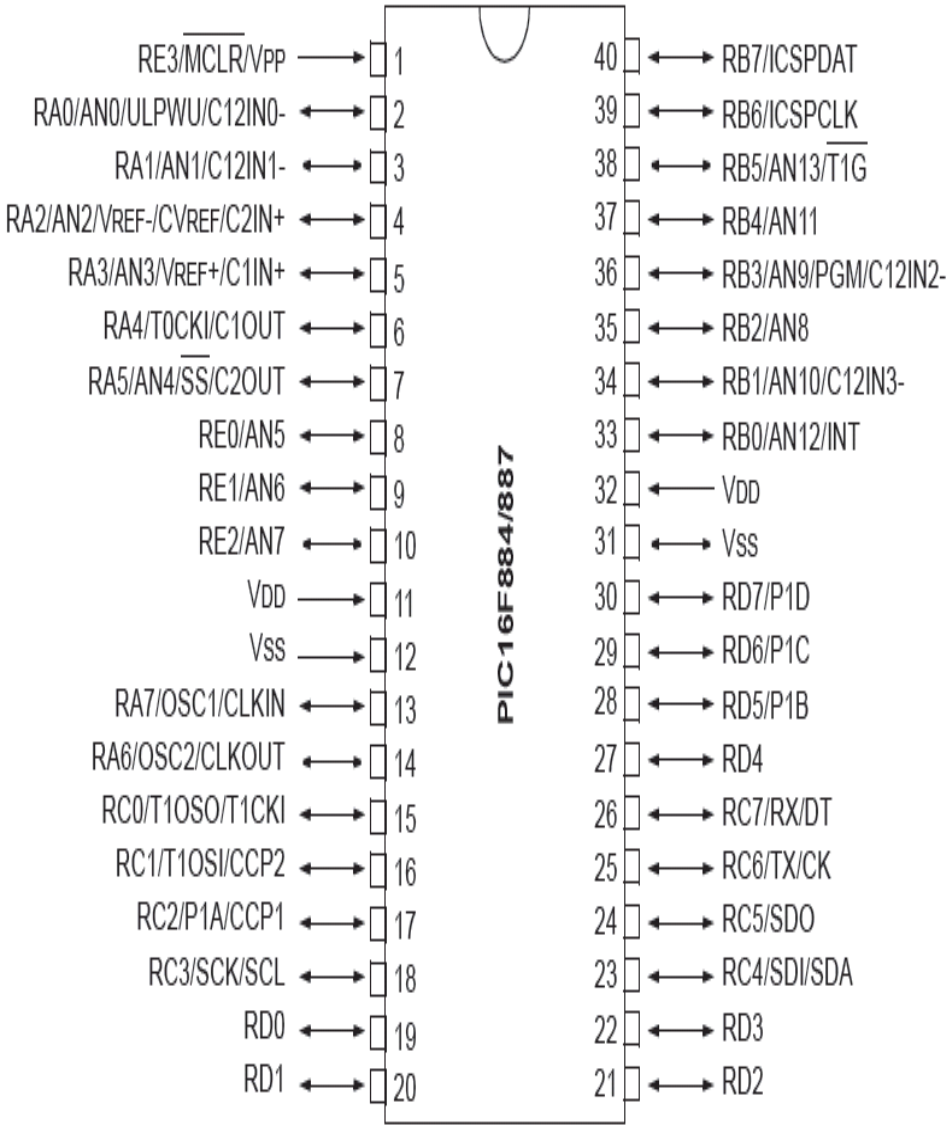


Fig.Pin diagram of PIC16F887

PIN DETAILS

I/O	Pin	Analog	Comparators	Timers	ECCP	EUSART	MSSP	Interrupt	Pull-up	Basic
RA0	2	AN0/ULPWU	C12IN0-	—	—	—	—	—	—	—
RA1	3	AN1	C12IN1-	—	—	—	—	—	—	—
RA2	4	AN2	C2IN+	—	—	—	—	—	—	VREF-/CVREF
RA3	5	AN3	C1IN+	—	—	—	—	—	—	VREF+
RA4	6	—	C1OUT	T0CKI	—	—	—	—	—	—
RA5	7	AN4	C2OUT	—	—	—	$\overline{SS}$	—	—	—
RA6	10	—	—	—	—	—	—	—	—	OSC2/CLKOUT
RA7	9	—	—	—	—	—	—	—	—	OSC1/CLKIN
RB0	21	AN12	—	—	—	—	—	IOC/INT	Y	—
RB1	22	AN10	C12IN3-	—	P1C	—	—	IOC	Y	—
RB2	23	AN8	—	—	P1B	—	—	IOC	Y	—
RB3	24	AN9	C12IN2-	—	—	—	—	IOC	Y	PGM
RB4	25	AN11	—	—	P1D	—	—	IOC	Y	—
RB5	26	AN13	—	$\overline{T1G}$	—	—	—	IOC	Y	—
RB6	27	—	—	—	—	—	—	IOC	Y	ICSPCLK
RB7	28	—	—	—	—	—	—	IOC	Y	ICSPDAT
RC0	11	—	—	T1OSO/T1CKI	—	—	—	—	—	—
RC1	12	—	—	T1OSI	CCP2	—	—	—	—	—
RC2	13	—	—	—	CCP1/P1A	—	—	—	—	—
RC3	14	—	—	—	—	—	SCK/SCL	—	—	—
RC4	15	—	—	—	—	—	SDI/SDA	—	—	—
RC5	16	—	—	—	—	—	SDO	—	—	—
RC6	17	—	—	—	—	TX/CK	—	—	—	—
RC7	18	—	—	—	—	RX/DT	—	—	—	—
RE3	1	—	—	—	—	—	—	—	Y(1)	$\overline{MCLR}$/VPP
—	20	—	—	—	—	—	—	—	—	VDD
—	8	—	—	—	—	—	—	—	—	VSS
—	19	—	—	—	—	—	—	—	—	VSS

Table.Pin Details of PIC16F887

ARCHITECTURE

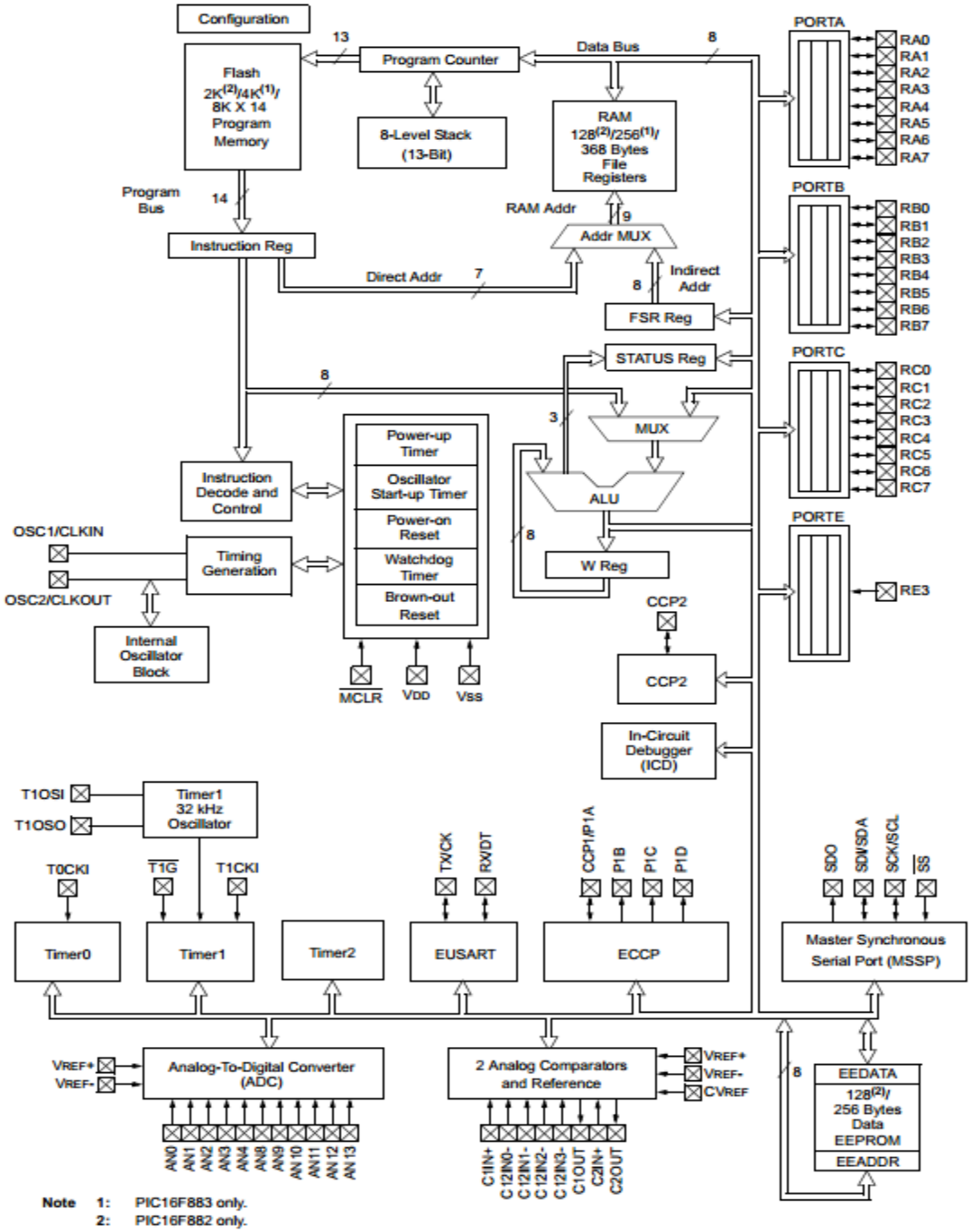


Fig.Architecture of PIC16F887

PIN DECRIPTION OF PIC16F887

Name	Function	Input Type	Output Type	Description
RA0/AN0/ULPWU/C12IN0-	RA0	TTL	CMOS	General purpose I/O.
	AN0	AN	—	A/D Channel 0 input.
	ULPWU	AN	—	Ultra Low-Power Wake-up input.
	C12IN0-	AN	—	Comparator C1 or C2 negative input.
RA1/AN1/C12IN1-	RA1	TTL	CMOS	General purpose I/O.
	AN1	AN	—	A/D Channel 1 input.
	C12IN1-	AN	—	Comparator C1 or C2 negative input.
RA2/AN2/VREF-/CVREF/C2IN+	RA2	TTL	CMOS	General purpose I/O.
	AN2	AN	—	A/D Channel 2.
	VREF-	AN	—	A/D Negative Voltage Reference input.
	CVREF	—	AN	Comparator Voltage Reference output.
	C2IN+	AN	—	Comparator C2 positive input.
RA3/AN3/VREF+/C1IN+	RA3	TTL	—	General purpose I/O.
	AN3	AN	—	A/D Channel 3.
	VREF+	AN	—	Programming voltage.
	C1IN+	AN	—	Comparator C1 positive input.
RA4/T0CKI/C1OUT	RA4	TTL	CMOS	General purpose I/O.
	T0CKI	ST	—	Timer0 clock input.
	C1OUT	—	CMOS	Comparator C1 output.
RA5/AN4/$\overline{SS}$/C2OUT	RA5	TTL	CMOS	General purpose I/O.
	AN4	AN	—	A/D Channel 4.
	$\overline{SS}$	ST	—	Slave Select input.
	C2OUT	—	CMOS	Comparator C2 output.
RA6/OSC2/CLKOUT	RA6	TTL	CMOS	General purpose I/O.
	OSC2	—	XTAL	Master Clear with internal pull-up.
	CLKOUT	—	CMOS	Fosc/4 output.
RA7/OSC1/CLKIN	RA7	TTL	CMOS	General purpose I/O.
	OSC1	XTAL	—	Crystal/Resonator.
	CLKIN	ST	—	External clock input/RC oscillator connection.
RB0/AN12/INT	RB0	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN12	AN	—	A/D Channel 12.
	INT	ST	—	External interrupt.
RB1/AN10/P1C/C12IN3-	RB1	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN10	AN	—	A/D Channel 10.
	P1C	—	CMOS	PWM output.
	C12IN3-	AN	—	Comparator C1 or C2 negative input.
RB2/AN8/P1B	RB2	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN8	AN	—	A/D Channel 8.
	P1B	—	CMOS	PWM output.

Legend: AN = Analog input or output; CMOS = CMOS compatible input or output; OD = Open Drain
TTL = TTL compatible input; ST = Schmitt Trigger input with CMOS levels
HV = High Voltage; XTAL = Crystal

Name	Function	Input Type	Output Type	Description
RB3/AN9/PGM/C12IN2-	RB3	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN9	AN	—	A/D Channel 9.
	PGM	ST	—	Low-voltage ICSP™ Programming enable pin.
	C12IN2-	AN	—	Comparator C1 or C2 negative input.
RB4/AN11/P1D	RB4	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN11	AN	—	A/D Channel 11.
	P1D	—	CMOS	PWM output.
RB5/AN13/$\overline{T1G}$	RB5	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN13	AN	—	A/D Channel 13.
	$\overline{T1G}$	ST	—	Timer1 Gate input.
RB6/ICSPCLK	RB6	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	ICSPCLK	ST	—	Serial Programming Clock.
RB7/ICSPDAT	RB7	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	ICSPDAT	ST	CMOS	ICSP™ Data I/O.
RC0/T1OSO/T1CKI	RC0	ST	CMOS	General purpose I/O.
	T1OSO	—	CMOS	Timer1 oscillator output.
	T1CKI	ST	—	Timer1 clock input.
RC1/T1OSI/CCP2	RC1	ST	CMOS	General purpose I/O.
	T1OSI	ST	—	Timer1 oscillator input.
	CCP2	ST	CMOS	Capture/Compare/PWM2.
RC2/P1A/CCP1	RC2	ST	CMOS	General purpose I/O.
	P1A	—	CMOS	PWM output.
	CCP1	ST	CMOS	Capture/Compare/PWM1.
RC3/SCK/SCL	RC3	ST	CMOS	General purpose I/O.
	SCK	ST	CMOS	SPI clock.
	SCL	ST	OD	I^2C™ clock.
RC4/SDI/SDA	RC4	ST	CMOS	General purpose I/O.
	SDI	ST	—	SPI data input.
	SDA	ST	OD	I^2C data input/output.
RC5/SDO	RC5	ST	CMOS	General purpose I/O.
	SDO	—	CMOS	SPI data output.
RC6/TX/CK	RC6	ST	CMOS	General purpose I/O.
	TX	—	CMOS	EUSART asynchronous transmit.
	CK	ST	CMOS	EUSART synchronous clock.
RC7/RX/DT	RC7	ST	CMOS	General purpose I/O.
	RX	ST	—	EUSART asynchronous input.
	DT	ST	CMOS	EUSART synchronous data.
RE3/$\overline{MCLR}$/VPP	RE3	TTL	—	General purpose input.
	$\overline{MCLR}$	ST	—	Master Clear with internal pull-up.
	VPP	HV	—	Programming voltage.
VSS	VSS	Power	—	Ground reference.
VDD	VDD	Power	—	Positive supply.

Fig. Pin Description of PIC16F887

ANALOG TO DIGITAL CONVERTER MODULE

When configuring and using the ADC the following functions must be considered. Port configuration

- Channel selection
- ADC voltage reference selection
- ADC conversion clock source
- Interrupt control
- Results formatting

Port configuration: The ADC can be used to convert both analog and digital signals. When converting analog signals, the I/O pin should be configured for analog by setting the associated TRIS and ANSEL bits. See the corresponding Port section for more information.

MEMORY ORGANIZATION

Program Memory Organization: The PIC16F882/883/884/886/887 has a 13-bit program counter capable of addressing a 2K x 14 (0000h-07FFh) for the PIC16F882, 4K x 14 (0000h-0FFFh) for the PIC16F883/PIC16F884, and 8K x 14 (0000h-1FFFh) for the PIC16F886/PIC16F887 program memory space. Accessing a location above these boundaries will cause a wraparound within the first 8K x 14 space.

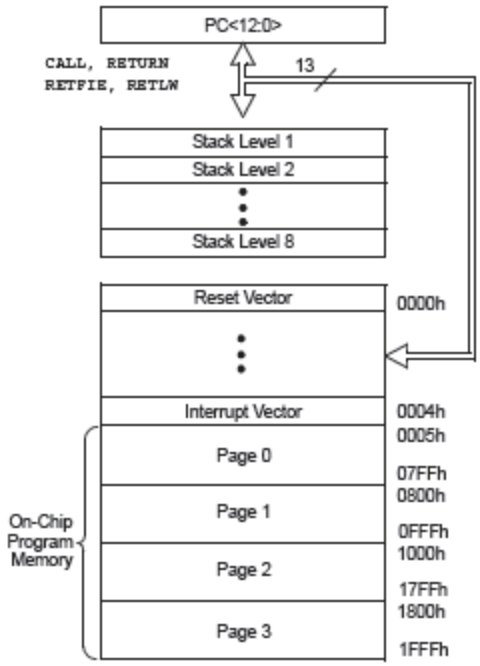


Figure 4.3 Program Memory Map and Stack

Data Memory Organization: The data memory is partitioned into four banks which contain the General Purpose Registers (GPR) and the Special Function Registers (SFR). The Special Function Registers are located in the first 32 locations of each bank. The General Purpose Registers, implemented as static RAM, are located in the last 96 locations of each Bank. Register locations F0h-FFh in Bank 1, 170h-17Fh in Bank 2 and 1F0h-1FFh in Bank 3, point to addresses 70h-7Fh in Bank 0. The actual number of General Purpose Resisters (GPR) implemented in each Bank depends on the device.

The STATUS register, shown in Register 2-1, contains:

- The arithmetic status of the ALU
- The Reset status
- The bank select bits for data memory (GPR and SFR)

The STATUS register can be the destination for any instruction, like any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic.

Furthermore, the TO and PD bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC(1)	C(1)
bit 7							bit 0

Fig. Status Register

Bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)

1 = Bank 2, 3 (100h-1FFh)

0 = Bank 0, 1 (00h-FFh)

Bit 6-5 **RP<1:0>:** Register Bank Select bits (used for direct addressing)

00 = Bank 0 (00h-7Fh)

01 = Bank 1 (80h-FFh)

10 = Bank 2 (100h-17Fh)

11 = Bank 3 (180h-1FFh)

Bit 4 **TO:** Time-out bit

1 = After power-up, CLRWDT instruction or SLEEP instruction

0 = A WDT time-out occurred

Bit 3 **PD:** Power-down bit

1 = After power-up or by the CLRWDT instruction

0 = By execution of the SLEEP instruction

Bit 2 **Z:** Zero bit

1 = The result of an arithmetic or logic operation is zero

0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit Carry/Borrow bit (ADDWF, ADDLW,SUBLW,SUBWF instructions)**(1)**

1 = A carry-out from the 4th low-order bit of the result occurred

0 = No carry-out from the 4th low-order bit of the result

bit 0 **C:** Carry/Borrow bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)**(1)**

1 = A carry-out from the Most Significant bit of the result occurred

0 = No carry-out from the Most Significant bit of the result occurred

3.2 ULTRASONIC SENSOR

PRODUCT FEATURES:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

(1) Using IO trigger for at least 10us high level signal,

(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.

(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.

Wire connecting direct as following:

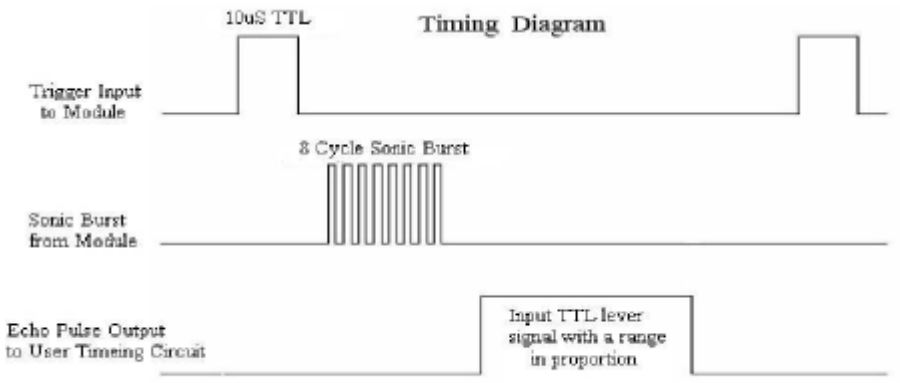
- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

Electric Parameter

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



Attention

The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.

When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

3.3 BLUETOOTH

Bluetooth is a wireless technology standard for exchanging data over short distances. Bluetooth uses a radio technology called frequency hopping spread spectrum, which chops up the data being sent and transmits chunks of it on up to 79 bands in the range 2400 -2483.5 MHz. This range is in the globally unlicensed Industrial, Scientific and Medical (ISM) 2.4 GHz short range radio frequency band.

Fig. Bluetooth module

Bluetooth is a packet based protocol with a master-slave structure. One master may communicate with up to7 slaves in a piconet. All devices share the master's clock. Packet exchange is based on the basic clock, defined by the master, which ticks at 312.5µs intervals. Two slots make up a slot pair of 1250 µs. In the simple case of single-slot packets the master transmits in even slots and receives in odd slots; the slave, conversely, receives in even slots and transmits in odd slots. Bluetooth provides a secure way to connect and exchange information between devices. It was principally designed as a low-bandwidth technology.

For transmission of data collected by the FSRs for feedback to the user the Bluetooth module is used in this project. EGBT-045MS and EGBT-046S Bluetooth Module are low cost replacements of our now retired EGBC-04 Bluetooth Module. EGBC-04 is an excellent Bluetooth Module, it is fully certified to Bluetooth standards, and is loaded with programmable features users had come to love. There is just one thing that went against it- it is expensive. It is easy to see why the EGBC-04 cost so much. Firstly, the manufacturer produced these specialty modules in relatively small volume; hence, there is no economy of the scale to speak of. Secondly, certification costs a lot of money; and this cost will have to be added on top of the manufacturing cost. Hence, EGBC-04 ended up costing about 10 times more expensive than its garden variety USB-type Bluetooth dongles cousins. Fortunately, at least one volume manufacturer have came up with an idea of producing a generic Bluetooth module in large quantity, for sale and distribution to developers who now have to put only the firmware functionalities.

This resulted in a huge drop in prices of these specialty Bluetooth modules, benefiting us experimenters and hobbyists. EGBT-045MS and EGBT-046S are generic Bluetooth Modules loaded with SPP firmware for UART wireless cable replacement functions. The EGBT- 045MS can be configured by the user to work either as a master or slave Bluetooth device using a set of AT commands. EGBT-046S, on the other hand, is permanently programmed as Bluetooth slave device. EGBT-046S, because of its simpler function, is a lot easier to use, and of course, costs less than EGBT-045MS. You can use it straight out of the box as a UART wireless cable replacement, without any need to add set-up codes in your microcontroller application firmware. Use the cheaper EGBT-046S if your application will connect to a master Bluetooth device, such as PC or laptops. Use the EGBT-045MS if your application must connect to a slave Bluetooth device, such as with EGBT-046S. Note that EGBT-045MS will work as well as a slave Bluetooth device.

Specification

- Radio Chip: CSR BC417
- Memory: External 8Mbit Flash
- Output Power: -4 to +6dbm Class 2
- Sensitivity: -80dbm Typical
- Bit Rate: EDR, up to 3Mbps
- Interface: UART
- Antenna: Built-in
- Dimension: 27W x 13H mm
- Voltage: 3.1 to 4.2VDC
- Current: 40mA max

3.4 TRANSFORMER

A **transformer** is a passive electrical device that transfers electrical energy from one electrical circuit to one or more circuits. A varying current in any one coil of the transformer produces a varying magnetic flux, which, in turn, induces a varying electromotive force across any other coils wound around the same core. Electrical energy can be transferred between the (possibly many) coils, without a metallic connection between the two circuits. Faraday's law of induction discovered in 1831 described the induced voltage effect in any coil due to changing magnetic flux encircled by the coil.

Transformers are used for increasing alternating voltages at low current (Step Up Transformer) or decreasing the alternating voltages at high current (Step Down Transformer) in electric power applications, and for coupling the stages of signal processing circuits.

Since the invention of the first constant-potential transformer in 1885, transformers have become essential for the transmission, distribution, and utilization of alternating current electric power. A wide range of transformer designs is encountered in electronic and electric power applications. Transformers range in size from RF transformers less than a cubic centimeter in volume, to units weighing hundreds of tons used to interconnect the power grid.

The main principle of operation of a transformer is mutual inductance between two circuits which is linked by a common magnetic flux. A basic transformer consists of two coils that are electrically separate and inductive, but are magnetically linked through a path of reluctance.

In short, a transformer carries the operations shown below:

- Transfer of electric power from one circuit to another.
- Transfer of electric power without any change in frequency.
- Transfer with the principle of electromagnetic induction.
- The two electrical circuits are linked by mutual induction.

DESCRIPTION

Good Quality Transformer, power supplies for all kinds of project & circuit boards. Step down 230 V AC to 9V with a maximum of 500mAmp current. Generally known as 9-0-9

Specification

- voltage: 2 x 9V
- current: 1 x 500mA
- rated power: 9VA

3.5 TTL

Transistor-Transistor Logic (TTL) Introduction to Logic Families: Logic Gates like NAND, NOR are used in daily applications for performing logic operations. The Gates are manufactured using semiconductor devices like BJT, Diodes or FETs. Different Gate's are constructed using Integrated circuits. Digital logic circuits are manufactured depending on the specific circuit technology or logic families. The different logic families are: 1. RTL(Resistor Transistor Logic) 2. DTL(Diode Transistor Logic) 3. TTL(Transistor Transistor Logic) 4. ECL(Emitter Coupled Logic) 5. CMOS(Complementary Metal Oxide Semiconductor Logic) Out of these RTL and DTL are rarely used. Features of Logic Families: 1. Fan Out: Number of loads the output of a GATE can drive without affecting its usual performance. By load we mean the amount of current required by the input of another Gate connected to the output of the given gate. 2. Power Dissipation: It represents

the amount of power needed by the device. It is measured in mW. It is usually the product of supply voltage and the amount of average current drawn when the output is high or low. 3. Propagation Delay: It represents the transition time which elapses when the input level changes. The delay which occurs for the output to make its transition is the propagation delay. 4. Noise Margin: It represents the amount of noise voltage allowed at the input, which doesn't affect the standard output. Introduction to TTLs: It is a logic family consisting completely of transistors. It employs transistor with multiple emitters. Commercially it starts with the 74 series like the 7404, 74S86 etc. It was build in 1961 by James L Bui and commercially used in logic design in 1963 Classification of TTL: TTLs are classified based on the output. 1. Open Collector Output: The main feature is that its output is 0 when low and floating when high. Usually an external Vcc may be applied. Transistor Q1 actually behaves as cluster of diodes placed back to back. With any of the input at logic low, corresponding emitter base junction is forward biased and the voltage drop across the base of Q1 is around 0.9V, not enough for the transistors Q2 and Q3 to conduct. Thus output is either floating or Vcc, i.e. High level.

3.6 RPS (REGULATED POWER SUPPLY)

Almost all electronic devices used in electronic circuits need a dc source of power to operate. The source of dc power is used to establish the dc operating points (Q-points) for the passive and active electronic devices incorporated in the system. The dc power supply is typically connected to each and every stage in an electronic system. It means that the single requirement common to all phases of electronics is the need for a supply of dc power. For portable low-power systems batteries may be used, but their operating period is limited. Thus for long time operation frequent recharging or replacement of batteries become much costlier and complicated. More frequently, however, electronic equipment is energized by a power supply, derived from the standard industrial or domestic ac supply by transformation, rectification, and filtering.(The combination of a transformer, a rectifier and a filter constitutes an ordinary dc power supply, also called an unregulated power supply).For many applications in electronics, unregulated power supply is not good enough because of the following reasons.

1. POOR REGULATION

The output voltage is far from constant as the load varies. The internal resistance of an ordinary power supply is relatively large (more than 30 ohms). So output voltage is significantly affected by the magnitude of current drawn from the supply. The voltage drop in the internal resistance of the supply increases directly with an increase in load current.

2. VARIATIONS IN THE AC SUPPLY MAINS

The permissible variation in the ac supply mains voltage as per Electricity Rules is 6% of its rated value. But in some countries, the variations in ac mains voltage is much more than this (sometimes it may vary from 180 V to 260 V). The dc output voltage being proportional to the input ac voltage, therefore, varies largely.

3. VARIATIONS IN TEMPERATURE

The dc output voltage varies with temperature, particularly if semiconductor devices are employed. Regulated power supply is an electronic circuit that is designed to provide a constant dc voltage of predetermined value across load terminals irrespective of ac mains fluctuations or load variations. Or in other words it converts unregulated AC into a constant DC. With the help of a rectifier it converts AC supply into DC. Its function is to supply a stable voltage (or less often current), to a circuit or device that must be operated within certain power supply limits. The output from the regulated power supply may be alternating or unidirectional, but is nearly always DC.

A regulated power supply essentially consists of an ordinary power supply and a voltage regulating device, as illustrated in the figure.

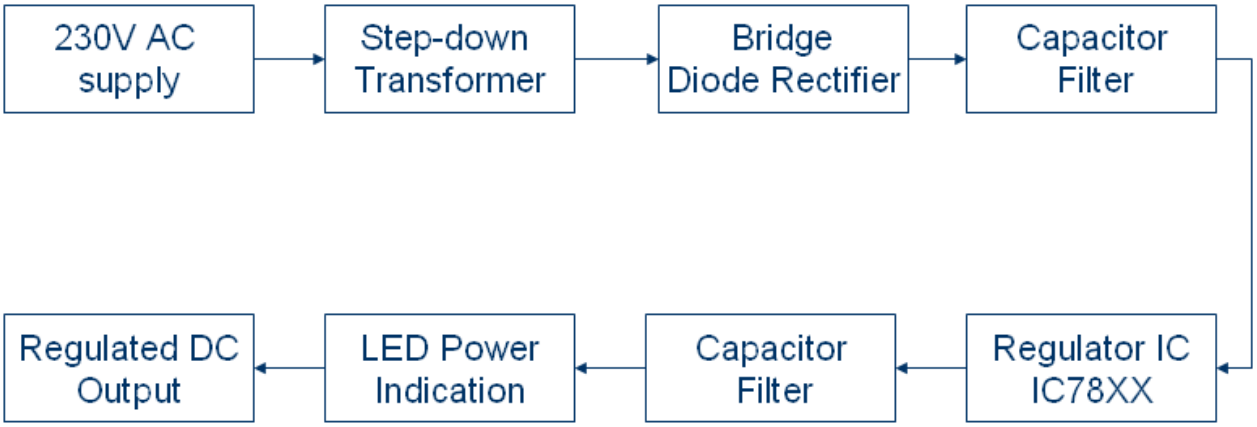


Fig. Block Diagram of RPS

The output from an ordinary power supply is fed to the voltage regulating device that provides the final output.

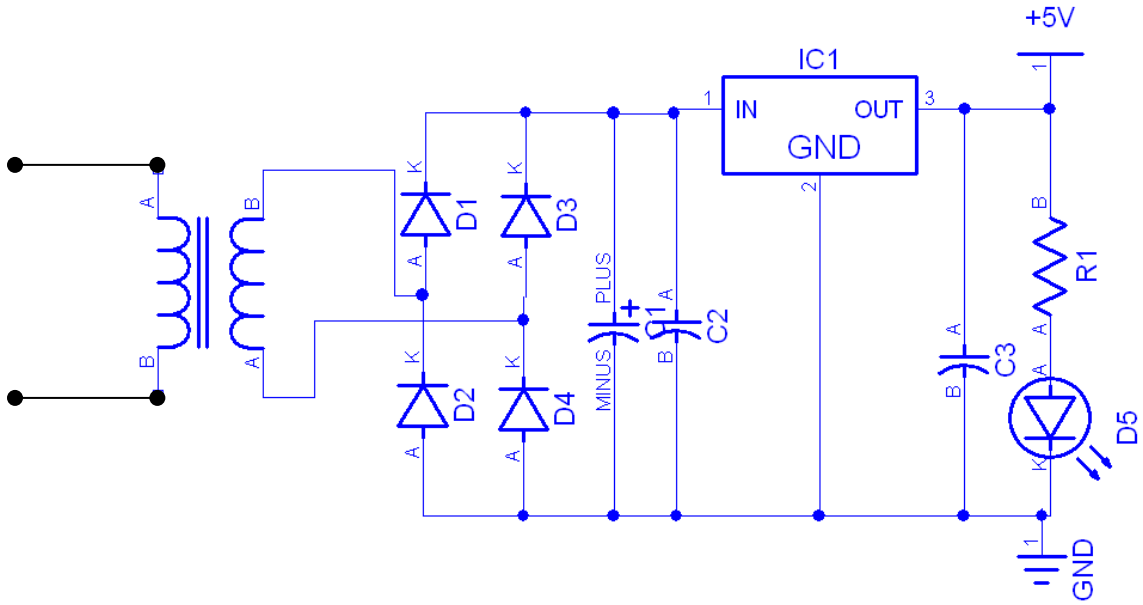


Fig.Regulated power supply circuit

The ac voltage, typically 230 V_{rms} is connected to a transformer which transforms that ac voltage to the level for the desired dc output. A bridge rectifier then provides a full-wave rectified voltage that is initially filtered by a ∏ (or C-L-C) filter to produce a dc voltage. The resulting dc voltage usually has some ripple or ac voltage variation.

A regulating circuit use this dc input to provide a dc voltage that not only has much less ripple voltage but also remains constant even if the input dc voltage varies somewhat or the load connected to the output dc voltage changes. The regulated dc supply is available across a voltage divider.

Graph of Regulated Power Supply

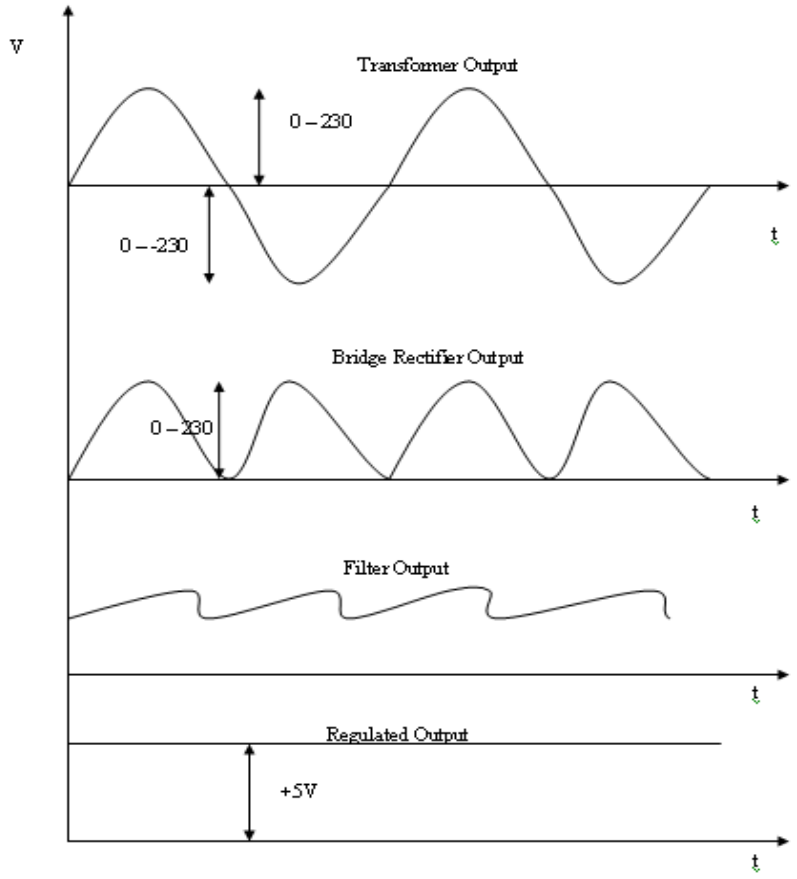


Fig.Graph of Regulated Power Supply

Often more than one dc voltage is required for the operation of electronic circuits. A single power supply can provide as many as voltages as are required by using a voltage (or potential) divider, as illustrated in the figure. As illustrated in the figure, a potential divider is a single tapped resistor connected across the output terminals of the supply. The tapped resistor may consist of two or three resistors connected in series across the supply. In fact, bleeder resistor may also be employed as a potential divider.

SUPPLY CHARACTERISTICS

There are various factors that determine the quality of the power supply like the load voltage, load current, voltage regulation, source regulation, output impedance, ripple rejection, and so on. Some of the characteristics are briefly explained below:

1. **LOAD REGULATION**

The load regulation or load effect is the change in regulated output voltage when the load current changes from minimum to maximum value.

- **Load regulation = Vno-load – Vfull-load**

Vno-load – Load Voltage at no load

Vfull-load – Load voltage at full load.

From the above equation we can understand that when Vno-load occurs the load resistance is infinite, that is, the out terminals are open circuited. Vfull-load occurs when the load resistance is of the minimum value where voltage regulation is lost.

% Load Regulation = [(Vno-load - Vfull-load)/Vfull-load] * 100

2. **MINIMUM LOAD RESISTANCE**

The load resistance at which a power supply delivers its full-load rated current at rated voltage is referred to as minimum load resistance.

Minimum Load Resistance = Vfull-load/Ifull-load

The value of Ifull-load, full load current should never increase than that mentioned in the data sheet of the power supply.

3. **SOURCE/LINE REGULATION**

In the block diagram, the input line voltage has a nominal value of 230 Volts but in practice, here are considerable variations in ac supply mains voltage. Since this ac supply mains voltage is the input to the ordinary power supply, the filtered output of the bridge rectifier is almost directly proportional to the ac mains voltage. The source regulation is defined as the change in regulated output voltage for a specified rage of lie voltage.

4. **OUTPUT IMPEDANCE**

A regulated power supply is a very stiff dc voltage source. This means that the output resistance is very small. Even though the external load resistance is varied, almost no change is seen in the load voltage. An ideal voltage source has an output impedance of zero.

5. **RIPPLE REJECTION**

Voltage regulators stabilize the output voltage against variations in input voltage. Ripple is equivalent to a periodic variation in the input voltage. Thus, a voltage regulator attenuates the ripple that comes in with the unregulated input voltage.

CHAPTER - 4

SOFTWARE PROFILE

PROTEUS 7.0 SIMULATION TOOL

Proteus 7.0 is a Virtual System Modeling (VSM) that combines circuit simulation, animated components and microprocessor models to co-simulate the complete microcontroller based designs. This is the perfect tool for engineers to test their microcontroller designs before constructing a physical prototype in real time.

This program allows users to interact with the design using on-screen indicators and/or LED and LCD displays and, if attached to the PC, switches and buttons. One of the main components of Proteus 7.0 is the Circuit Simulation -- a product that uses a SPICE3f5 analogue simulator kernel combined with an event-driven digital simulator that allow users to utilize any SPICE model by any manufacturer. Proteus VSM comes with extensive debugging features, including breakpoints, single stepping and variable display for a neat design prior to hardware prototyping. In summary, Proteus 7.0 is the program to use when we want to simulate the interaction between software running on a microcontroller and any analog or digital electronic device connected to it.

4.1 CCS COMPILER

A compiler is a computer program (or set of programs) that transforms source code written in a programming language (the source language) into another computer language (the target language, often having a binary form known as object code). The most common reason for wanting to transform source code is to create an executable program.

This integrated C development environment gives developers the capability to quickly produce very efficient code from an easily maintainable high level language. The compiler includes built-in functions to access the PIC microcontroller hardware such as READ_ADC to read a value from the A/D converter. Discrete I/O is handled by describing the port characteristics in a PROGRAM. Functions such as INPUT and OUTPUT_HIGH will properly maintain the tri-state registers. Variables including structures may be directly mapped to memory such as I/O ports to best represent the hardware structure in C.

CCS C COMPILER FEATURES

- Built in libraries that work with all chips for RS232 serial I/O, I2C, discrete I/O and precision delays.
- Integrates with MPLAB IDE and other simulators and editors for source level debugging. Standard HEX file and debug files ensure compatibility with all programmers.
- Formatted printf allows easy formatting and display in HEX or decimal.
- Efficient function implementation allows call trees deeper than the hardware stack.
- Source code drivers included for LCD modules, keypads, 24xx and 94xx serial EEPROM's, X10, DS1302 and NJU6355 real time clocks, Dallas touch memory devices, DS2223 and PCF8570 serial SRAM, LTC1298 and PCF8591 A/D converters, temperature sensors, digital pots, I/O expander and much more.
- Access to hardware features from easy to use C functions, timers, A/D, EEPROM, SSP, PSP, USB, I2C and more.
- 1, 8, 16 and 32 bit integer types and 32 bit floating point.
- Assembly code may be inserted anywhere in the source and may reference C variables.
- Automatic linking handles multiple code pages.
- Inline functions supported to save stack space; Linker will automatically determine the best architecture or it can be manually specified.
- Compiler directives determine if tri-state registers are refreshed on every I/O or if the I/O is as fast as possible.

4.2 ECLIPSE

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including Ada, ABAP, C, C++, C#, COBOL, D, Fortran, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Rust, Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to

develop documents with LaTeX (via a TeXlipse plug-in) and packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

The initial codebase originated from IBM VisualAge. The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Since the introduction of the OSGi implementation (Equinox) in version 3 of Eclipse, plug-ins can be plugged-stopped dynamically and are termed (OSGI) bundles.

Eclipse software development kit (SDK) is free and open-source software, released under the terms of the Eclipse Public License, although it is incompatible with the GNU General Public License. It was one of the first IDEs to run under GNU Classpath and it runs without problems under IcedTea.

4.3 EMBEDDED C PROGRAMMING

Looking around, we find ourselves to be surrounded by various types of embedded systems. Be it a digital camera or a mobile phone or a washing machine, all of them has some kind of processor functioning inside it. Associated with each processor is the embedded software. If hardware forms the body of an embedded system, embedded processor acts as the brain, and embedded software forms its soul. It is the embedded software which primarily governs the functioning of embedded systems.

During infancy years of microprocessor based systems, programs were developed using assemblers and fused into the EPROM's. There used to be no mechanism to find what the program was doing. LEDs, switches, etc. were used to check correct execution of the program. Some 'very fortunate' developers had In-circuit Simulators (ICEs), but they were too costly and were not quite reliable as well.

As time progressed, use of microprocessor-specific assembly-only as the programming language reduced and embedded systems moved onto C as the **embedded programming**

language of choice. C is the most widely used programming language for embedded processors/controllers. Assembly is also used but mainly to implement those portions of the code where very high timing accuracy, code size efficiency, etc. are prime requirements.

Initially C was developed by Kernighan and Ritchie to fit into the space of 8K and to write (portable) operating systems. Originally it was implemented on UNIX operating systems. As it was intended for operating systems development, it can manipulate memory addresses. Also, it allowed programmers to write very compact codes. This has given it the reputation as the language of choice for hackers too.

As assembly language programs are specific to a processor, assembly language didn't offer portability across systems. To overcome this disadvantage, several high level languages, including C, came up. Some other languages like PLM, Modula-2, Pascal, etc. also came but couldn't find wide acceptance. Amongst those, C got wide acceptance for not only embedded systems, but also for desktop applications. Even though C might have lost its sheen as mainstream language for general purpose applications, it still is having a strong-hold in embedded programming. Due to the wide acceptance of **C in the embedded systems**, various kinds of support tools like compilers & cross-compilers, ICE, etc. came up and all this facilitated development of **embedded systems using C**.

SOURCE CODE

4.4 SOURCE CODE

```
#include <16F887.h>

#define SEL_key portb_7
#define busstop portb_6
#define office portb_5
#define home portb_4

unsigned char user_loc[15],urt_rcv=0,rcv_cnt=0;
unsigned long lcd_dis_count = 0;

unsigned long adc_val2;
unsigned long adc_map2;

unsigned long sensor_scan = 0;
unsigned long cur_loc=0;
//unsigned char home[2] ="h";
//unsigned char office[2] ="o";
//unsigned char bus_stop[2] ="b";
unsigned long needed_loc = 0;
int8 ob_flag=0,Page_flip=0,Loc_flag=0;
short F_flag=0,B_flag=0,L_flag=0,R_flag=0;
unsigned long Set_obstacle_F=0,Set_obstacle_B=0,Set_obstacle_R=0,Set_obstacle_L=0;
```

```
short R_place_Flag = 1;
short  s_flag=0;

void adc2_scan()
{
   unsigned long ran1,ran_count1=0;
   ran1=0;
    for(ran_count1=0;ran_count1<=9;ran_count1++)
    {
      ran1 +=(unsigned long)read_adc(); //* 0.1035;
      delay_us(30);
    }

   adc_val2 = (ran1/10) ;
   if ((adc_val2 > 400) && (adc_val2 < 500))
   {
    cur_loc  =1;
   }
   else if ((adc_val2 > 650) && (adc_val2 < 800))
   {
    cur_loc  =3;
   }
   else if (adc_val2 > 800)
   {
    cur_loc  =2;
```

```
   }
}

void Distance_condition()
{
  if ((distance_F) < (Set_obstacle_F + 2)) F_flag=1;
  else F_flag=0;
  if ((distance_B) < (Set_obstacle_B + 2)) B_flag=1;
  else B_flag=0;
  if ((distance_R) < (Set_obstacle_R + 2)) R_flag=1;
  else R_flag=0;
  if ((distance_L) < (Set_obstacle_L + 2)) L_flag=1;
  else L_flag=0;
}

void Distance_condition_check()
{
  if(F_flag==0 && B_flag==0 && R_flag==0 && L_flag==0) ob_flag =0;// no obstacle
  if(F_flag==1 && B_flag==1 && R_flag==1 && L_flag==1) ob_flag =1;// ALL SIDE

  if(F_flag==1 && B_flag==0 && R_flag==0 && L_flag==0) ob_flag =2;// Front
  if(F_flag==0 && B_flag==0 && R_flag==1 && L_flag==0) ob_flag =3;// right
  if(F_flag==0 && B_flag==0 && R_flag==0 && L_flag==1) ob_flag =4;// left
  if(F_flag==0 && B_flag==1 && R_flag==0 && L_flag==0) ob_flag =5;// Back
```

```
  if(F_flag==1 && B_flag==1 && R_flag==0 && L_flag==0) ob_flag =6;// Front & Back
  if(F_flag==1 && B_flag==0 && R_flag==1 && L_flag==0) ob_flag =7;// Front & right
  if(F_flag==1 && B_flag==0 && R_flag==0 && L_flag==1) ob_flag =8;// Front & left
  if(F_flag==0 && B_flag==1 && R_flag==1 && L_flag==0) ob_flag =9;// Back & right
  if(F_flag==0 && B_flag==1 && R_flag==0 && L_flag==1) ob_flag =10;// Back & Left
  if(F_flag==0 && B_flag==0 && R_flag==1 && L_flag==1) ob_flag =11;// left & right

  if(F_flag==1 && B_flag==1 && R_flag==1 && L_flag==0) ob_flag =12;// Front, Back & right
  if(F_flag==1 && B_flag==1 && R_flag==0 && L_flag==1) ob_flag =13;// Front, Back & Left
  if(F_flag==0 && B_flag==1 && R_flag==1 && L_flag==1) ob_flag =14;// Front, Right & Left
  if(F_flag==0 && B_flag==1 && R_flag==1 && L_flag==1) ob_flag =15;// Back, Right & Left

}



void Blue_transmit()
{
  if(ob_flag==1)
  {
    fprintf(,"A"); // ALL sides
    lcd_write(0x80,0);
    printf(lcd_write_string," Obstacle Detected    ");
    lcd_write(0xc0,0);
    printf(lcd_write_string,"    All Sides    ");
    delay_ms(1000);
```

```
}
if(ob_flag==2)
{
  fprintf("B"); // Obstacle Front side
  lcd_write(0x80,0);
  printf(lcd_write_string," Obstacle Detected   ");
  lcd_write(0xc0,0);
  printf(lcd_write_string,"   Front Side   ");
  delay_ms(1000);
}
if(ob_flag==3)
{
  fprintf("C"); // Obstacle Right side
  lcd_write(0x80,0);
  printf(lcd_write_string," Obstacle Detected   ");
  lcd_write(0xc0,0);
  printf(lcd_write_string,"   Right Side   ");
  delay_ms(1000);
}
if(ob_flag==4)
{
  fprintf("D"); // obstacle Left side
  lcd_write(0x80,0);
  printf(lcd_write_string," Obstacle Detected   ");
  lcd_write(0xc0,0);
```

```
    printf(lcd_write_string,"    Left Side    ");
    delay_ms(1000);
  }
  if(ob_flag==5)
  {
    fprintf("E"); // obstacle Back side
    lcd_write(0x80,0);
    printf(lcd_write_string," Obstacle Detected    ");
    lcd_write(0xc0,0);
    printf(lcd_write_string,"    Back Side    ");
    delay_ms(1000);
  }
  if(ob_flag==6)
  {
    fprintf("F"); // obstacle Front & Back side
    lcd_write(0x80,0);
    printf(lcd_write_string," Obstacle Detected    ");
    lcd_write(0xc0,0);
    printf(lcd_write_string," Front & Back Side    ");
    delay_ms(1000);
  }
  if(ob_flag==7)
  {
    fprintf("G"); // obstacle Front & Right side
    lcd_write(0x80,0);
```

```
    printf(lcd_write_string," Obstacle Detected    ");
    lcd_write(0xc0,0);
    printf(lcd_write_string," Front & Right Side    ");
    delay_ms(1000);
  }
  if(ob_flag==8)
  {
    fprintf("H"); // obstacle Front & Left side
    lcd_write(0x80,0);
    printf(lcd_write_string," Obstacle Detected    ");
    lcd_write(0xc0,0);
    printf(lcd_write_string," Front & Left Side    ");
    delay_ms(1000);
  }
  if(ob_flag==9)
  {
    fprintf("I"); // obstacle Back & Right side
    lcd_write(0x80,0);
    printf(lcd_write_string," Obstacle Detected    ");
    lcd_write(0xc0,0);
    printf(lcd_write_string," Back & Right Side    ");
    delay_ms(1000);
  }
  if(ob_flag==10)
  {
```

```
    fprintf("J");// obstacle Back & Left side
    lcd_write(0x80,0);
    printf(lcd_write_string," Obstacle Detected    ");
    lcd_write(0xc0,0);
    printf(lcd_write_string," Back & Left Side    ");
    delay_ms(1000);
  }
  if(ob_flag==11)
  {
    fprintf("K"); // obstacle Left & Right side
    lcd_write(0x80,0);
    printf(lcd_write_string," Obstacle Detected    ");
    lcd_write(0xc0,0);
    printf(lcd_write_string," Left & Right Side    ");
    delay_ms(1000);
  }
  if(ob_flag==12)
  {
    fprintf("L");// obstacle Front,Back & Right side
    lcd_write(0x80,0);
    printf(lcd_write_string," Obstacle Detected    ");
    lcd_write(0xc0,0);
    printf(lcd_write_string,"Front, Back & Right Side");
    delay_ms(1000);
  }
```

```
if(ob_flag==13)
{
  fprintf("M");// obstacle Front,Back & Left side
  lcd_write(0x80,0);
  printf(lcd_write_string," Obstacle Detected    ");
  lcd_write(0xc0,0);
  printf(lcd_write_string,"Front, Back & Left Side");
  delay_ms(1000);
}
if(ob_flag==14)
{
  fprintf("N");// obstacle Front,Right & Left side
  lcd_write(0x80,0);
  printf(lcd_write_string," Obstacle Detected    ");
  lcd_write(0xc0,0);
  printf(lcd_write_string,"Front, Right & Left Side");
  delay_ms(1000);
}
if(ob_flag==15)
{
  fprintf("O");// obstacle Back,Right & Left side
  lcd_write(0x80,0);
  printf(lcd_write_string," Obstacle Detected    ");
  lcd_write(0xc0,0);
  printf(lcd_write_string,"Back, Right & Left Side");
```

```
    delay_ms(1000);
  }
}


void Left()
{
    lcd_write(0x80,0);
    printf(lcd_write_string,"Turn            ");
    lcd_write(0xc0,0);
    printf(lcd_write_string,"Left          ");
  fprintf(USB,"#");
  delay_ms(100);
  fprintf(USB,"Q");
 delay_ms(2000);



}

void Right()
{
    lcd_write(0x80,0);
    printf(lcd_write_string,"Turn            ");
    lcd_write(0xc0,0);
    printf(lcd_write_string,"Right         ");
```

```
  fprintf(USB,"#");
  delay_ms(1000);
  fprintf(USB,"P");
  delay_ms(1000);
  fprintf(USB,"$");
  delay_ms(2000);

}



void stright()
{
    lcd_write(0x80,0);
    printf(lcd_write_string,"Go straight        ");
    lcd_write(0xc0,0);
    printf(lcd_write_string,"                ");
  fprintf(USB,"#");
  delay_ms(1000);
  fprintf(USB,"R");
  delay_ms(1000);
  fprintf(USB,"$");
   delay_ms(2000);

}
```

```
void R_Home()
{
        if (needed_loc == cur_loc)
        {
        lcd_write(0x80,0);
        printf(lcd_write_string,"    Reached      ");
        lcd_write(0xc0,0);
        printf(lcd_write_string,"     Home        ");
        R_place_Flag = 1;
     fprintf(USB,"#");
     delay_ms(1000);
     fprintf(USB,"U");
     delay_ms(1000);
     fprintf(USB,"$");
      delay_ms(2000);
     Distance_condition_check();
     Blue_transmit();
       }

}

void R_Office()
{
```

```
        if (needed_loc == cur_loc)
        {
        Loc_flag=0;
        lcd_write(0x80,0);
        printf(lcd_write_string,"    Reached      ");
        lcd_write(0xc0,0);
        printf(lcd_write_string,"     Office       ");
        R_place_Flag = 1;
    fprintf(USB,"#");
    delay_ms(1000);
    fprintf(USB,"S");
    delay_ms(1000);
    fprintf(USB,"$");
     delay_ms(2000);
       }
}

void R_Bus_stop()
{
    if (needed_loc == cur_loc)
    {
    Loc_flag=0;
    lcd_write(0x80,0);
    printf(lcd_write_string,"     Reached    ");
```

```
      lcd_write(0xc0,0);
      printf(lcd_write_string,"     Bus Stop    ");
      R_place_Flag = 1;
   fprintf(USB,"#");
   delay_ms(1000);
   fprintf(USB,"T");
   delay_ms(1000);
   fprintf(USB,"$");
    delay_ms(2000);
      }

}



void Blu_condition_check()
{
    //if(s_flag)
    //{
        delay_ms(100);
        if(Loc_flag==1) //// Home
        {
          lcd_write(0x80,0);
          printf(lcd_write_string,"User Location      ");
          lcd_write(0xc0,0);
          printf(lcd_write_string,"Home              ");
```

```
delay_ms(1000);
needed_loc = 1;
//fprintf(USB,"Loc : %lu\n\r",cur_loc);
  if (cur_loc == 1 && ob_flag ==0)
  {
    delay_ms(10000);
    lcd_write(0x80,0);
    printf(lcd_write_string,"   You are          ");
    lcd_write(0xc0,0);
    printf(lcd_write_string,"  Home Only            ");
      fprintf(USB,"#");
      delay_ms(1000);
      fprintf(USB,"X");
      delay_ms(1000);
      fprintf(USB,"$");
      Loc_flag=0;
    delay_ms(10000);

  }
  if (cur_loc == 2 && ob_flag ==0)
  {
    L_R_R();
      R_Home();
  }
  if (cur_loc == 3 && ob_flag ==0)
```

```
    {
      R_L_R();
        R_Home();
    }
}
if(Loc_flag==2) //// Office
{
  lcd_write(0x80,0);
  printf(lcd_write_string,"User Location     ");
  lcd_write(0xc0,0);
  printf(lcd_write_string,"Office         ");
  needed_loc = 2;
  delay_ms(1000);
    if (cur_loc == 1 && ob_flag ==0)
    {
      L_R_R();
        R_Office();
    }
    if (cur_loc == 2 && ob_flag ==0)
    {
        delay_ms(10000);
      lcd_write(0x80,0);
      printf(lcd_write_string,"   You are         ");
      lcd_write(0xc0,0);
      printf(lcd_write_string,"  Office Only           ");
```

```
            fprintf(USB,"#");
            delay_ms(1000);
            fprintf(USB,"W");
            delay_ms(1000);
            fprintf(USB,"$");
            Loc_flag=0;
          delay_ms(10000);
       }
       if (cur_loc == 3 && ob_flag ==0)
       {
         L_R_L();
            R_Office();
       }
  }
  if(Loc_flag==3)  //// Bus stop
  {
    lcd_write(0x80,0);
    printf(lcd_write_string,"User Location      ");
    lcd_write(0xc0,0);
    printf(lcd_write_string,"Bus stop      ");
    needed_loc = 3;
    delay_ms(1000);
      if (cur_loc == 1 && ob_flag ==0)
      {
        L_L_R();
```

```
        R_Bus_stop();
    }
    if (cur_loc == 2 && ob_flag ==0)
    {
      R_L_R();
        R_Bus_stop();
    }
    if (cur_loc == 3 && ob_flag ==0)
    {

       delay_ms(10000);
      lcd_write(0x80,0);
      printf(lcd_write_string,"   You are          ");
      lcd_write(0xc0,0);
      printf(lcd_write_string,"  Bus stop Only    ");
        fprintf(USB,"#");
        delay_ms(1000);
        fprintf(USB,"V");
        delay_ms(1000);
        fprintf(USB,"$");
        Loc_flag=0;
      delay_ms(10000);

    }
}
```

```
        //s_flag=0;
        //clear_urt();
    //}
}



void welcome_display()
{
     lcd_write(0x80,0);
     printf(lcd_write_string,"  SMART TALKING    ");
     lcd_write(0xc0,0);
     printf(lcd_write_string,"      GLASS       ");
}

void location_display()
{
  if(Page_flip == 0)
     {
        lcd_clear();
        lcd_write(0x80,0);
        printf(lcd_write_string,"Obstacle_F:%02lucm ",distance_F);
        lcd_write(0xc0,0);
        printf(lcd_write_string,"SET     :%02lucm" ,Set_obstacle_F);
```

```
  }

else if(Page_flip == 1)
  {
    lcd_clear();
    lcd_write(0x80,0);
    printf(lcd_write_string,"Obstacle_B:%02lucm   ",distance_B);
    lcd_write(0xc0,0);
    printf(lcd_write_string,"SET    :%02lucm",Set_obstacle_B);
  }


else if(Page_flip == 2)
  {
    lcd_clear();
    lcd_write(0x80,0);
    printf(lcd_write_string,"Obstacle_R:%02lucm   ",distance_R);
    lcd_write(0xc0,0);
    printf(lcd_write_string,"SET    :%02lucm",Set_obstacle_R);
  }

else if(Page_flip == 3)
  {
    lcd_clear();
    lcd_write(0x80,0);
```

```
    printf(lcd_write_string,"Obstacle_L:%02lucm   ",distance_L);
    lcd_write(0xc0,0);
    printf(lcd_write_string,"SET    :%02lucm",Set_obstacle_L);

  }

else if(Page_flip == 4)
  {
     lcd_clear();
     lcd_write(0x80,0);
     printf(lcd_write_string,"My Location");
     lcd_write(0xc0,0);
      if (cur_loc == 1)
        printf(lcd_write_string,"Home          ");
      else if (cur_loc == 2)
        printf(lcd_write_string,"Office        ");
      else if (cur_loc == 3)
        printf(lcd_write_string,"Bus Stop      ");
  }

  Page_flip++;
  if(Page_flip>5)
  {
    Page_flip = 0;
  }
```

```
}

void main()
{
  while (1)
  {
    adc2_scan();
    ultra_trigger_routine_F();
    ultra_trigger_routine_B();
    ultra_trigger_routine_R();
    ultra_trigger_routine_L();
    Distance_condition();
    Distance_condition_check();
    Blu_condition_check();

    if (lcd_dis_count > 800)
    {
      location_display();
      lcd_dis_count = 0;
    }

    if(sensor_scan > 5000)
    {
      Blue_transmit();
```

```
      sensor_scan = 0;
    }

    if(home == 0)
    {
      delay_ms(50);
      Loc_flag = 1;
    }

    if(office == 0)
    {
      delay_ms(50);
      Loc_flag = 2;
    }

    if(busstop == 0)
    {
      delay_ms(50);
      Loc_flag = 3;
    }

  }
}
```

CHAPTER 5

CONCLUSION

5. CONCULSION

A novel navigation system is designed and implemented which helps blind people to navigate safely. PIC microcontroller was used to develop the smart obstacle detection system which allows the blind person to avoid obstacles using the feedback through voice. The primary objective of this design was to make the system cost effective and easier to handle for a visually impaired person. They can easily walk with android mobile and the ultrasonic sensor will simply detect the obstacles and help the persons to maneuver around it. This system provides voice feedback alert to the user if any obstacle is around and within 70cm. The ultrasonic sensor is adjusted in a way to ensure user convenience in detecting obstacles in four directions. The voice alert continuously informs user about the obstacle until the user moves away from the obstacle within the range of 70cm. This project suggested that this aid will be an effective, low-cost and user friendly solution for navigation problems of visually impaired person.

CHAPTER 6

REFERENCES

6. REFERENCES

[1] A. D. Bagdanov, A. Del Bimbo, L. Seidenari, and L. Usai ,“Real-time hand status recognition from RGB-D imagery,” In Proceedings of the 21st International Conference on Pattern Recognition (ICPR ’12), pp. 2456–2459, November 2012.

[2] M. Elmezain, A. Al-Hamadi, and B. Michaelis, “A robust method for hand gesture segmentation and recognition using forward spotting scheme in conditional random fields,” in Proceedings of the 20th International Conference on Pattern Recognition (ICPR ’10), pp. 3850–3853, August 2010.

[3] C.-S. Lee, S. Y. Chun, and S. W. Park, “Articulated hand configuration and rotation estimation using extended torus manifold embedding,” in Proceedings of the 21st International Conference on Pattern Recognition (ICPR ’12), pp. 441–444, November 2012.

[4] M. R. Malgireddy, J. J. Corso, S. Setlur, V. Govindaraju, and D. Mandalapu, “A framework for hand gesture recognition and spotting using sub-gesture modeling,” in Proceedings of the 20th International Conference on Pattern Recognition (ICPR ’10), pp. 3780–3783, August 2010.

[5] P. Suryanarayan, A. Subramanian, and D.Mandalapu, “Dynamic hand pose recognition using depth data,” in Proceedings of the 20th International Conference on Pattern Recognition (ICPR ’10), pp. 3105– 3108, August 2010.

[6] S. Park, S. Yu, J. Kim, S. Kim, and S. Lee, “3D hand tracking using Kalman filter in depth space,” Eurasip Journal on Advances in Signal Processing, vol. 2012, no. 1, article 36, 2012.

[7] J. L. Raheja, A.Chaudhary, andK. Singal, “Tracking of fingertips and centers of palm using KINECT,” in Proceedings of the 2nd International Conference on Computational Intelligence, Modelling and Simulation (CIMSim’11), pp. 248–252, September 2011.

[8] Y. Wang, C. Yang, X. Wu, S. Xu, and H. Li, “Kinect based dynamic hand gesture recognition algorithm research,” in Proceedings of the 4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC ’12), pp. 274–279, August 2012.

[9] R. Yang, S. Sarkar, and B. Loeding, “Handling movement epenthesis and hand segmentation ambiguities in continuous sign language recognition using nested dynamic programming,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 3, pp. 462–477, 2010.

[10] Z. Zafrulla, H. Brashear, T. Starner, H. Hamilton, and P.Presti, “American sign language recognition with the kinect,” in Proceedings of the 13th ACM International Conference on Multimodal Interfaces (ICMI '11), pp. 279–286, November 2011.

[11] D.Uebersax, J. Gall,M.Van denBergh, and L. VanGool, “Real time sign language letter and word recognition”, from depth ,The Scientific World Journal 9 data in Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV ’11), pp. 383–390,November 2011

CHAPTER 7
COST ANALYSIS

7. COST ANALYSIS

S.NO.	COMPONENT	COST
1.	PIC MIC – 16F887	2250
2.	IC	1200
3.	TTL	550
4.	4 X ULTRA SONIC SENSOR	1000
5.	TRANSFORMER	300
6.	POTENTIO METER	350
7.	BLUETOOTH	450
8.	POWER SOURCE	500
9.	LCD PANEL	300
10.	PANEL BOARD	300
11.	APPLICATION	1200
12.	CONNECTING WIRES	100
	TOTAL	**8500**

.