#### EMBEDDED SYSTEM

#### INDUSTRIAL TRAINING REPORT

for the course

18ECP102L- Industrial Training- I

Submitted by

Pushpal Das (**RA1911004010565**)

Semester – V

Academic Year: 2020–21

*In partial fulfilment for the award of the degree* 

Of

#### **BACHELOR OF TECHNOLOGY**

in

# ELECTRONICS AND COMMUNICATION ENGINEERING of FACULTY OF ENGINEERING AND TECHNOLOGY



S.R.M. Nagar, Kattankulathur, Kancheepuram District **AUGUST 2020** 

1

#### SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

### **BONAFIDE CERTIFICATE**

This is to certify that this report titled "EMBEDDED SYSTEM" is the Bonafide work of "PUSHPAL DAS (RA1911004010565)", who carried out the industrial training. Certified further, that to the best of my knowledge the work reported herein does

not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Signature Signature

Mr. Pushpal Das Mrs.G.Kalaimagal ECE III Year Industrial training coordinator Dept. of Electronics and Dept. of Electronics and Communication Engineering

Communication Engineering

2

### **ACKNOWLEDGEMENT**

The success and final outcome of this Project required a lot of guidance, assistance from many Professors and I am extremely privileged to have got this all along the completion of my project. I owe my deep gratitude to my training guide Mr. Tushar Khurmi who took keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a practical approach. I am thankful and fortunate enough to get constant encouragement, support and guidance from all Professors of ECE Department. I would like to express my gratitude towards my parents and seniors for their kind co-operation and encouragement which helped me in completion of the Project.



### **ABSTRACT**

The objectives of MSME-DI, Agra, are to provide Technical Consultancy Services to the entrepreneurs in addition to Industrial Management Trainings, Entrepreneurs Development Programmes, Short Term Management Trainings for Micro, Small & Medium Entrepreneurs, Industrial Motivational Campaigns, Implementation and Monitoring of PMEGP Scheme, Entrepreneurship & Skilled

Development Programmes, Establishing of Sub Contracting Exchange, Awareness on Energy Conservation, Quality Control & it's up gradation and Ancillary Development. The Institute also provides Market Information, Industrial Potential Survey Report,

Statistical Information, Identification of Thrust Industries, Export Promotion, Directory of MSMEs, Economic Information to set up MSMEs.

### CONTENT

- Arduino Programming
- Proteus Designing
- Embedded C in Keil
- 8051, ARM
- Projects on Microcontroller

### ARDUINO

- A small computer on a single chip containing a processor, memory, and input/output. Typically, "embedded" inside some device that they control. A microcontroller is often small and low cost
- Development board-A printed circuit board designed to facilitate work with a particular microcontroller.

Typical components include-

- power circuit
- programming interface
- basic input; usually buttons and LEDs

## WHAT IS THE AURDINO?

Arduino provides open source electronies prototyping platforms based on flexible, easy-to-use hardware and software. Arduino prototyping platforms are intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments. Arduino's prototyping platforms can sense the environment by receiving input from a variety of sensors and can affect their surroundings by controlling lights, motors, and other actuators. Arduino projects can be stand-alone or they can communicate with software running on a computer.

### LET'S GET STARTED!

- Download & install the Arduino environment (IDE)
- Connect the board to your computer via the UBS.
- Cable.
- If needed, install the drivers (not needed in lab).
- Launch the Arduino IDE
- Select your board.
- Select your serial port.
- Open the blink example.
- Upload the program.

## A Little Bit About Programming

- For controlling the Arduino board and performing computations.
- Digital I/O

- digitalRead() digitalWrite() pinMode()
- Analog I/O

analogRead(),

\_analogReference() analogWrite()

#### Zero, Due & MKR Family

 analogReadResolution() analogWriteResolution()

#### Advanced I/O

noTone()

pulseIn()

pulseInLong()

shiftIn()

shiftOut()

tone()

#### Time

delay()

delayMicroseconds()

micros()

abs()

constrain()

map()

max()

min()

pow()

sq() sqrt()

#### Trigonometry

cos()

sin()

tan()

#### Characters

isAlpha()

isAlphaNumeric()

isAscii()

isControl()

isDigit()

isGraph()

isHexadecimalDigit()

isLowerCase()

isPrintable()

isPunct()

isSpace()

isUpperCase()

isWhitespace()

Functions

For controlling the Arduino board and performing computations.

#### Random Numbers

random()randomSeed()

#### Bits and Bytes

• bit()

bitClear()

bitRead()

bitSet()

bitWrite()

highByte()

lowByte()

#### External Interrupts

attachInterrupt()detachInterrupt()

#### Interrupts

interrupts()noInterrupts()

#### Communication

SerialStream

#### USB

Keyboard Mouse

Arduino data types and constants.

#### Constants

• HIGH LOW

INPUT OUTPUT INPUT\_PULLUP

LED\_BUILTIN

true false

Floating Point Constants

Integer Constants

#### Conversion

(unsigned int)

(unsigned long)

byte()

char()

float()

int()

long()

word()

#### Variable Scope & Qualifiers

const

scope

static

volatile

#### Utilities

```
PROGMEM
sizeof()
```

Variables

Arduino data types and constants.

```
Data Types
```

array

bool

boolean

byte

char

double

float

int

long

short size\_t

string

String()

unsigned char

unsigned int

unsigned long

void

word

Structure

The elements of Arduino (C++) code.



#### Sketch

loop()

setup()

#### Control Structure

break

continue

do...while

else

for

goto

if

return switch...case

while

#### Further Syntax

```
#define (define)
```

#include (include)

/\* \*/ (block comment)

// (single line comment)

; (semicolon)

{} (curly braces)

#### millis()

Structure

The elements of Arduino (C++) code.



#### Arithmetic Operators

% (remainder)

- \* (multiplication)
- + (addition)
- (subtraction)
- / (division)
- = (assignment operator)

#### **Comparison Operators**

```
!= (not equal to)
```

- < (less than)
- <= (less than or equal to)
- == (equal to)
- > (greater than)
- >= (greater than or equal to)

#### **Boolean Operator**

! (logical not)

&& (logical and)

|| (logical or)

• Structure

The elements of Arduino (C++) code.



#### Pointer Access Opera

& (reference operator)

\* (dereference operator)

#### **Bitwise Operators**

& (bitwise and)

<< (bitshift left)

>> (bitshift right)

^ (bitwise xor)

(bitwise or)

~ (bitwise not)

#### **Compound Operators**

%= (compound remainder)

&= (compound bitwise and)

\*= (compound multiplication)

++ (increment)

+= (compound addition)

-- (decrement)

-= (compound subtraction)

/= (compound division)

^= (compound bitwise xor)

|= (compound bitwise or)

### **OUR FIRST PROGRAM**

Blink
Turns on an LED on for one second, then off for one second, repeatedly.

Most Arduinos have an on-board LED you can control. On the Uno and Leonardo, it is attached to digital pin 13. If you're unsure what pin the on-board LED is connected to on your Arduino model, check

// the setup function runs once when you press reset or power the board void setup() {
 // initialize digital pin 13 as an output.
 pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
 digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
 delay(1000); // wait for a second
 digitalWrite(13, LOW); // turn the LED off by making the voltage LOW

**TERMINOLOGY** 

delay(1000);

- "Sketch"-A program which we can write to run it on an Arduino Board.
- "pin"-An input or output connected to something (example-LED, input from a nob)
- "digital"-Value is either high or low(aka on/off, one/zero) switch state.
- "analog"-value ranges usually from 0-255(example LED Brightness, motor speed.)

### DIGITAL? ANALOG?

anaiog pins can aiso pe useu as Digital IO

- Digital has two values: on and off
- Analog has many (infinite) values
- Computers don't really do analog, they quantize
- We should always remember the 6 analog input pins---voltage, analog signal, digital point, digital signal, sampled point, time.

## COMPUTER SYSTEM

A Computer system is a programmable machine that receive input, stores and manipulate data/ information and provide output.

**BLOCK DIAGRAM** 

Basic computer system consists of Central Processing Unit (CPU), Memory (RAM and ROM), input / output (I/O) Unit.

### CENTRAL PROCESSING UNIT

- The portion of a computer system that carries out the instruction of a computer program.
- The primary element carrying out the computer's function .It is the unit that reads and executes program instructions.
- The data in the instruction tells the processor what to do.

### **MEMORY**

- Physical device used to store data or Program on a temporary or permanent basis for use in an electronic digital computer.
- Computer main memory comes in to principles variation.
- (1) Random Access Memory (RAM)
- (2) Read Only Memory (ROM).
  - RAM can be read and write to anytime the CPU command's it, but ROM is preloaded with data and software that never changes, so the CPU can only read from it.
  - RAM is typically use to store the computer's initial start-up instruction.
  - In general, the computers of RAM are erased when power to the computers is turned off, but ROM retains its data after turned off computer.

## INPUT/OUTPUT UNIT

- Input /Output (I/O), refers to the communication between an information processing system (such as computer) and
  outside world possibly a human or another information processing system.
- Inputs are the signal or data received by the system and Output are the signal or data sent from it. Device that provides input or Outputs to the computers are called Peripherals.

### **DATA SIZE**

### DATA SIZL

- BIT = 1 or 0
- NIBBLE = 4 BIT
- BYTE = 8 BIT
- WORD = 16 BIT
- LONG WORD = 32 BIT

### EMBEDDED SYSTEM

- Special-purpose computer system designed to perform a dedicated function.
- Performs one or a few pre-defined tasks, usually with very specific requirements, and often includes task-specific hardware and mechanical parts not usually found in a general-purpose computer.
- COMPONENTS OF EMBEDDED SYSTEM

#### **Analog Components**

- Sensors (LDR, LM35 etc.)
- Digital Components

- Processor, Coprocessors
- Memories
  - O Controllers, Buses
- Application Specific Integrated Circuits
- Converters A2D, D2A.
- Software
- Application Programs
  - EMBEDDED DESIGNING
- Core hardware
- Firmware tools
- Programming tools
- Hardware tools

#### DESIGNING

- Core hardware
- 8051 Microcontroller
- PIC (Peripheral Interface Control) Microcontroller
- AVR Microcontroller
- ARM (Advance risk machine) Microcontroller
- FPGA (Field Programmable Gate Array)
- CPLD (Complex Programmable Logic Device)
- Firmware tools
- **8051-** KEIL Compiler & Assembler
- PIC Source Boost, micro c
- AVR Code Vision Avr
- ARM KEIL advanced versions
- Programming tools
- 8051 Flash Magic, ECE Flash
- PIC Matrix pppv3, pick it
- AVR Pony Prog
- ARM Flash Utility
- Hardware tools
- Hardware programmers
- Application debuggers
- EMBEDDED APPLICATIONS AUTOMOBILES: Fuel Injection control (for fuel efficiency), Air bags and Automatic braking (for safety), and car entertainment systems.
- MEDICAL ELECTRONICS: Many sophisticated medical instruments (Body Scanners, Heart rate monitors, Pacemaker etc)
   Industrial Control: such as CNC machines are example embedded systems.
- BUSINESS APPLICATIONS: Vending machines, scanners, printers.
  - CONSUMER ELECTRONICS: Cameras, Toys, Cellular Phones, Washing Machines
- AVIONICS: Airplanes, Satellite Stations
- Defense: RADARs, SONARs (for surveillance), Guided Missile Systems

### **MICROCOMPUTERS**

- Microprocessor is not a complete microcomputer it contains only processing unit.
- A microprocessor is a single chip semiconductor device also which is a computer on chip, but not a complete computer.
- Its CPU contains an ALU, a program counter, a stack pointer, some working register, a clock timing circuit and interrupt circuit on a single chip.
- To make complete microcomputer, one must add memory usually ROM and RAM, memory decoder, an oscillator and a number of serial and parallel ports.

#### MICROCONTROLLER-

- A microcontroller is a functional computer system-on-a-chip. It contains a processor, memory, and programmable input/output peripherals.
- Microcontrollers include an integrated CPU, memory (a small amount of RAM, program memory, or both) and peripherals capable of input and output.

### CLASSIFICATION OF MICROCONTRLLER

- Classification According to Number of Bits
- In 8-bit microcontroller, the point when the internal bus is 8-bit then the ALU is performs the arithmetic and logic operations. The examples of 8-bit microcontrollers are Intel 8031/8051.
- The **16-bit** microcontroller performs greater precision and performance as compared to 8-bit. 16-bit microcontroller can proceed data up to 16 bits.
- The 32-bit microcontroller uses the 32-bit instructions to perform the arithmetic and logic operations.
- Fx-ARM7

#### Classification According to Memory Devices

- Embedded memory microcontroller: When an embedded system has a microcontroller unit that has all the
  functional blocks available on a chip is called an embedded microcontroller. For example, 8051 having program &
  data memory, I/O ports, serial communication, counters and timers and interrupts on the chip is an embedded
  microcontroller.
- External Memory Microcontroller: When an embedded system has a microcontroller unit that has not all the functional blocks available on a chip is called an external memory microcontroller. For example, 8031 has no program memory on the chip is an external memory microcontroller.

#### Classification According to Instruction Set

- CISC: CISC is a Complex Instruction Set Computer. It allows the programmer to use one instruction in place of many simpler instructions.
- RISC: The RISC is stands for Reduced Instruction set Computer, this type of instruction sets reduces the design of microprocessor for industry standards. It allows each instruction to operate on any register or use any addressing mode and simultaneous access of program and data.
- Classification According to
- Memory Architecture
- Harvard Memory Architecture Microcontroller: The point when a microcontroller unit has a dissimilar memory
  address space for the program and data memory, the microcontroller has Harvard memory architecture in the
  processor.
- Princeton Memory Architecture Microcontroller: The point when a microcontroller has a common memory address
  for the program memory and data memory, the microcontroller has Princeton memory architecture in the processor.
- CRITERIA FOR CHOOSING MICROCONTROLLER
- 1. The first and most criteria in choosing a microcontroller is that it must meet the task and cost effective.
- 2.In analysing the need of microcontroller base project we first see whether an 8-bit, 16 bit, 32-bit Microcontroller can be best handling the computing need.
- 3. Speed (what is the highest speed that microcontroller support).
- 4. Power Consumption: This is specially criteria for battery power product.
- 5. The amount of RAM and ROM in chip.
- 6.The no. of I/O pins and the timer on the chip.
- 7. How easily it is to upgrade to high performance.

Cost per unit: - This is important in term of final cost of the product in which a microcontroller is used.

INTODUCTION TO MICROCONTROLLER

In 1981 intel corporation introduced an 8-bit microcontroller called 8051. The microcontroller had 128 byte of RAM ,4 kb of ROM on chip, 2 Timers, one serial port and four ports (each 8 bit) on a single chip at the time it will also refer as system on chip. The 8051 this is an 8-bit microcontroller, means CPU can work on only 8 bits at a time.

• Data larger than 8 bits as to broken into 8-bit piece to be processed by CPU.

The 8051 has total of four I/O ports each of 8 bits. Although the 8051 can have a maximum of 64 kilobytes of on chip ROM, many manufacturers have put only 4 k byte on chip. The 8051 become widely popular after intel allowed other manufacturer to make a market any flavour of 8051.

- 8051 Family Members
- 8031 no ROM
- 80xx mask ROM
- 87xx EPROM
- 89xx Flash EEPROM
- 89xx
- 8951
- 8952
- 8953
- 8955
- 898252
- 891051
- 892051
- Example (AT89C51, AT89LV51)
- AT= ATMEL(Manufacture)
- C = CMOS technology
- LV= Low Power (3.0v)
- Comparison some of the 8051 Family Members
- 8051 -4k 128 2. 8031 128 2. 8751 4k eprom 128 2.
- 8052 8k from 256 3. 8032 256 3.
- 8752 8k eprom 256 3.

## 8051 Basic Component

- 128 bytes internal RAM
- Four 8-bit I/O ports (P0 P3).
- Two 16-bit timers/counters
- One serial interface
- 64k external memory for code
- 64k external memory for data
- 210 bits addressable
- The basic 8051 Core
- 8-bit CPU optimized for control applications
- Capability for single bit Boolean operations.
- Supports up to 64K of program memory.
- Supports up to 64K of data memory.
- 4 K bytes of on-chip program memory.
- Newer devices provide more.
- 128 bytes of on-chip data RAM
- Four 8-bit ports.
- Two 16-bit timer/counters
- UART
- Interrupts
- On-chip clock oscillator

## **Block Diagram**

External interrupts

On-chip

ROM RAM Timer

Timer/Counter

Interrupt Control

CPU

ROM for program code

Bus

On-chip RAM

4 I/O Ports

Timer 1 Timer 0

Serial

Counter Inputs

Address/Data

## Special Function Registers (SFR)

Special function registers are part of RAM memory. Their purpose is predefined by the manufacturer and cannot be changed therefore.

P0 P1 P2 P3 TxD RxD

Since their bits are physically connected to particular circuits within the microcontroller, such as A/D converter, serial communication module etc.

Any change of their state directly affects the operation of the microcontroller or some of the circuits.

#### Program Counter

Program Counter is an engine running the program and points to the memory address containing the next instruction to execute.

After each instruction execution, the value of the counter is incremented by 1.

For this reason, the program executes only one instruction at a time.

However...the value of the program counter can be changed at any moment, which causes a "jump" to a new memory location.

After jumping, the counter resumes even and monotonous automatic counting +1, +1, +1...

## Central Processor Unit (CPU)

- As its name suggests, this is a unit which monitors and controls all processes within the microcontroller and the user cannot affect its work.
- Instruction decoder :-

Is a part of the electronics which recognizes

program instructions and runs other circuits on the basis of that.

The abilities of this circuit are expressed in the

"instruction set" which is different for each

- Arithmetical Logical Unit (ALU):- Performs all mathematical and logical operations upon data.
- Accumulator is an SFR closely related to the operation of ALU. It is a kind of working desk used for storing all data upon which some operations should be executed. It also stores the results ready for use in further processing.

## Memory Organization

- 8051 Memory Organization
- The total memory of 8051 is logically divided into
- 1) Program

Memory

- 2) Data Memory
- 1) Program

Memory

The 8051 has 4k byte of internal program memory (on chip memory-ROM) Which starts from 0000H to 0FFFH .The address of memory location is 16 bit and normally stored in program counter to fetch instruction from memory. It is used to store program only not data.

The program memory of 8051 can be increased by Interfacing 64KB of external memory ROM i.e. program which address starts From 0000H to FFFFH.

2.Data

Memory

It is used for temporary storage of data .It is divided into-

- 1) Internal data memory
- 2) External data memory

\_Internal data memory

It is consist of two part the first is the RAM block of 128 bytes & second is set of address from 80H to FFH. Which is used for special function registers.

External data

memory

The external RAM of 64KB can be connected.

### Pin out Description

- Pins 1-8: Port 1 Each of these pins can be configured as an input or an output.
- Pin 9: RSA logic one on this pin disables the microcontroller and clears the contents of most registers. In other words, the positive voltage on this pin resets the microcontroller. By applying logic zero to this pin, the program starts execution from the beginning.
- Pins 10-17: Port 3 Similar to port 1, each of these pins can serve as general input or output. Besides, all of them have alternative functions:
- 22-03-2016 37
- Pin 10: RXD Serial asynchronous communication input or Serial synchronous communication output.
- Pin 11: TXD Serial asynchronous communication output or Serial synchronous communication clock output.
- Pin 12: INT0 Interrupt 0
- input.

- Pin 13: INT1 Interrupt 1
- input.
- Pin 14: T0 Counter 0 clock
- input.
- Pin 15: T1 Counter 1 clock
- Pin 16: Write to external (additional)
- RAM
- Pin 17: RD Read from external RAM
- Pin 18, 19: X2, X1 Internal oscillator input.
- A quartz crystal which specifies operating frequency is usually connected to these pins.
- Pin 20: GND Ground
- Pin 21-28: Port 2 If there is no intention to use external memory then These port pins are configured as general inputs/outputs. In case external memory is used, the higher address byte, i.e. addresses A8-A15 will appear on this port.
- Pin 29: PSEN (program store enable)If external ROM is used for storing program every time the microcontroller reads a byte from memory.
- It is a program store enable Pin whenever device is executing code from external program memory PSEN is active ( logic 0). It is used to fetch the code from external memory .
- For internal program memory this pin should be inactive(logic 1 or high).
- Pin 30: ALE (Address latch enable) 8051 port -0 provides both address & data. So we have to make separate .
- The ALE pin is used for de-multiplexing the address &data by connecting to the G pin (enable pin) of the 74LS373 latch IC. When ALE=0 data path is selected & when ALE=1 address path is selected.
- Pin 31: EA It is stand for external access . It is an input pin and must be connected to either Vcc or Ground.
- By applying logic zero (ground) to this pin, P2 and P0 are used for data and address transmission.
- By connecting Vcc to indicate that the program code is stored in the microcontroller's on chip ROM.

External data memory

Pin 32-39: Port 0 Similar to P2, if external memory is not used, these pins can be used as general inputs/outputs. Otherwise, P0 is configured as address output (A0-A7) when the ALE pin is driven high (1) or as data output (Data Bus) when the ALE pin is driven low (0).

Pin 40: VCC +5V power

supply.

#### Pull-Up Resistors

• Port 0 as both input and output port A each pin must connected externally to a 10k-ohm pull up register. This is due to the fact that p0 is an open drain .

#### \_Register B

It is used to store 8 bit data. It is a special function register.

Register B is used with accumulator for multiplication and  $\,$  division.

#### Rn Register

• This is common name for the total 8 general purpose registers (R0, R1, R2,....R7).

They are not true SFR similar to Accumulator they are used for temporary storing variable/value and intermediate result .

These registers are stored in four bank.

DPTR

#### Data Pointer Register

- It is a 16 bit registers but it is divided into two registers named DPH & DPL.
- DPTR does not have a single internal address it can be used as 16 -bit register or as two independent 8 bit register.
- The DPTR is used to furnish memory address and external data access.

**STACK** 

The stack is a section of RAM used by the CPU to store information temporarily. This information could be data or an address. The CPU needs this storage area since there are only a limited number of register.

- The Register used to access the stack is called the SP(stack pointer)
- register. The stack pointer in the 8051 is only 8 bit wide which means
- that it can take a value of 00 to FFH.
- Stack Pointer
- It is a 8 bit register. It is used to hold an internal RAM address i.e. top of the stack.
- It indicates the last byte push on the stack.
- During stack write operation (push) on the stack pointer is first incremented by 1 & than save the data on to the stack. After resetting the **m**c the stack pointer is initialized with 07H address which cause the stack pointer to begin from location 08H.

## Analog To Digital converter (ADC)

- An analog-to-digital converter is a device which converts continuous signals to discrete digital numbers. The reverse operation is performed by a digital-to-analog converter (DAC).
- Typically, an ADC is an electronic device that converts an input analog voltage (or current) to a digital number proportional to the magnitude of the voltage or current.

## <u>Steps to interface 0809 ADC with 8051 uc</u>

- 1. Set initial values of SOC,EOC and OE.
- 2. Set ADD A, ADD B,ADD C.
- 3. High to low pulse on ALE.
- 4. High to low pulse on SOC with clock.
- 5. Monitor EOC bit.
- 6. Send h to l pulse to OE.
- 7. Read data.

#include<reg51.h>

sbit EOC=P1^3;

- 8. Repeat the steps.
- Program to get data from ADC

void main(void)

{

SOC=0;

OE=0;

EOC=1;

P2=0xff;

while(1)

{

sbit ALE=P1^2;

sbit SOC=P1^1;

sbit OE=P1^0;

```
sbit SET0=P1^4;
sbit SET1=P1^5;
sbit SET2=P1^6;
sbit CLOCK=P1^7;
void clock(void);
void clock(void)
int b;
for(b=0;b<=30000;b++)
CLOCK=~CLOCK;
void delay(int x);
SET0=1;SET1=1; SET2=1; //for the select line
  ALE=1; SOC=1;
  clock();
  SOC=0; ALE=0;
  clock();
  while(EOC==0);
  OE=1;
  P3=P2;
```

Baud Rates in the 8051

The 8051 transfers and receives data serially at many different baud rates by using UART.

- UART divides the machine cycle frequency by 32 and sends it to Timer 1 to set the baud rate.
- Signal change for each roll over of timer Timer 1, mode 2 (8-bit, auto-reload)
- Define TH1 to set the baud rate.
  - O XTAL = 11.0592 MHz
  - O The system frequency = 11.0592 MHz / 12 = 921.6 kHz O Timer 1 has 921.6 kHz/32 = 28,800 Hz as source.

  - O TH1=FDH means that UART sends a bit every 3 timer source.
  - O Baud rate = 28,800/3= 9,600 Hz

## ರ್ಡ್ಡ್ಯ regarding serial communication.

- Serial control register: SCON
- SM0, SM1 Serial port mode specified

- REN (Receive enable) set/cleared by software to enable/disable reception.
- TI Transmit interrupt flag.
- RI Receive interrupt flag.
- SM2 = TB8 = TB8 = 0 (not widely used)
- SM0, SM1
- SM1 and SM0 determine the framing of data.
  - O SCON.6 (SM1) and SCON.7 (SM0)
  - O Only mode 1 is compatible with COM port of PC.
- TB8, RB8
- TB8:The 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.
- RB8: In modes 2 and 3, the 9th data bit that was received. In Mode 1, if SM2 = 0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.
- REN (Receive Enable)
- SCON.4
- Set/cleared by software to enable/disable reception.
  - O REN=1
    - If we want the 8051 to both transfer and receive data, REN must be set to 1.
  - O REN=0
    - The receiver is **disabled**.
    - The 8051 can not receive data

#### Transfer Data with the TI flag

- The following sequence is the steps that the 8051 goes through in transmitting a character via TxD:
  - a. The byte character to be transmitted is written into the SBUF register.
  - b. It transfers the start bit.
  - c. The 8-bit character is transferred one bit at a time.
  - d. The stop bit is transferred.

Steps to transmit data from 8051 to pc

- 1.Load SCON register.
- 2.Laod TMOD register(select timer1 mode 2).
- 3. Load baud rate in TH1.
- 4. Start the timer.
- 5. Place character in SBUF.
- 6. Monitor TI flag.
- 7. Clear TI flag.

TR1=1;

### Program To Transmit Data To PC

```
#include<reg51.h>
unsigned char arr[]="Advance Technology";
void serial_init()
{
TMOD=0x20;
SCON=0x50;
TH1=0xfd;
```

```
TI=0;
}
void serial_trans(unsigned char trans)
SBUF=trans;
while(TI==0);
TI=0;
}
void main ()
{
Int i;
serial_init();
  while(1)
     for(i=0;i<=17;i++)
serial_trans(arr[i]);
    }
}
     • Steps to Receive data
   from pc to 8051
1.Load SCON register.
2.Laod TMOD register(select timer1 mode 2).
3. Load baud rate in TH1.
4. Start the timer.
5. Monitor RI flag.
7. Read SBUF.
8. Clear RI.
Program to Received data form PC
#include<reg51.h>
void serial_init()
{
```

```
TMOD=0x20;
SCON=0x50;
TH1=0xfd;
TR1=1;
TI=0;
}
unsigned char serial_rec()
while(RI==0);
RI=0;
    return(SBUF);
}
void main ()
  serial_init();
   while(1)
  {
     P0=serial_rec();
   }
}
```

ROBOTICS
"A robot is a reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks."

- O Speed
- O It can work hazardous/dangerous environment
- O To perform repetitive task
- O Efficiency
- 0 Accuracy
- Adaptability 0
- Automation vs. Robots
  - Automation Machinery designed to carry out a specific task 0
  - Bottling machine 0
  - Dishwasher
  - 0 Paint sprayer
  - 0
  - - Robots machinery designed
- to carry out a variety of tasks
  - O Pick and place arms
  - O Mobile robots
  - 0 Computer Numerical Control
  - 0 machines
- Types of Robot
  - O Wheeled

- O Legged
- O Climbing
- O Flying

## Degrees of freedom

#### Control

- Open loop, i.e., no feedback, deterministic
- Closed loop, i.e., feedback, maybe a sense of touch and/or vision

#### Degrees of freedom

- Degrees of freedom—number of independent motions
  - O Translation—3independent directions
  - O Rotation-- 3 independent axes
  - O 2D motion = 3 degrees of freedom: 2 translation, 1 rotation
  - O 3D motion = 6 degrees of freedom: 3 translation, 3 rotation.

#### Types of motors used in robotics

- O DC motors
  - DC non geared motors
  - DC geared motors
- O Stepper motors
- O Servo motors

## Interfacing of 8051 with DC Motors

- The most common actuator in mobile robotics
- simple, cheap, and easy to use.
- come in a great variety of sizes, to
- accommodate different robots and tasks.
- As the name suggests, a motor which uses a DC (Direct Current) power, can run in both directions, speed Controllable.
- Principles of Operation
- DC motors convert electrical into mechanical energy.
- They consist of permanent magnets and loops of wire inside.
- When current is applied, the wire loops generate a magnetic field, which reacts against the outside field of the static magnets.
- The interaction of the fields produces the movement of the shaft/armature.
- Thus, electromagnetic energy becomes motion.
- DC Motor Working
- Direction of rotation controlled by polarity of current / voltage
- Speed of rotation controlled by average energy (power) fed to the motor

### **PWM**

- Pulse width modulation is a technique for reducing the amount of power delivered to a DC motor.
- Instead of reducing the voltage operating the motor (which would reduce its power), the motor's power supply is rapidly switched on and off.
- The percentage of time that the power is on determines the percentage of full operating power that is accomplished.
- Program to run dc motor with PWM

•

#include<reg51.h>

#include<intrins.h>

sbit dc\_motor = P0^0;

void delay(int num)

```
int a,b;
for(b=0; b<num; b++)
  for(a=0;a<=140;a++)
_nop_();
}
void PWM(unsigned char Ton)
{
dc_motor=1; // motor ON
delay(Ton);
dc_motor=0; // motor OFF
delay(100-Ton);
}
void main()
{
while(1)
{
PWM(50);
}
}</pre>
```

## lnterfacing of 8051with Stepper motor

Used for measured rotation

m1=0;m2=1;m3=0;m4=0;

- Can be held at a particular position of the shaft
- Ideal for many autonomous robots requiring higher precision
- Stepper Motor Working
- Stepping Sequences for Single Coil Excitation
- Only one coil is active at a given instant of time
- Program to control the direction of stepper motor

```
#include<reg51.h>
sbit m1=P0^0;
sbit m2=P0^1;
sbit m3=P0^2;
sbit m4=P0^3;
void delay(unsigned int x)
{
unsigned int a;
for(a=0;a<=x;a++);</li>
}
void motor_clk() // motor moves in clock wise direction
{
m1=1;m2=0;m3=0;m4=0;
delay(30000);
```

```
delay(30000);
    m1=0;m2=0;m3=1;m4=0;
    delay(30000);
    m1=0;m2=0;m3=0;m4=1;
    delay(30000);
Direction, steps of stepper motor
Program to control the direction, steps and speed of stepper motor
#include<reg51.h>
sbit m1=P0^0;
sbit m2=P0^1;
sbit m3=P0^2;
sbit m4=P0^3;
unsigned int a,b;
void delay(unsigned int x)
{
for(a=0;a<=x;a++);
void motor_clk(int y) // motor moves in clock wise direction
for(a=0; a<=y; a++)
    m1=1;m2=0;m3=0;m4=0;
delay(30000);
    m1=0;m2=1;m3=0;m4=0;
    delay(30000);
    m1=0;m2=0;m3=1;m4=0;
    delay(30000);
    m1=0;m2=0;m3=0;m4=1;
    delay(30000);
   }
}
void motor_anticlk(int z)
                            // motor moves in anticlock wise direction
for(a=0; a<=z; a++)
m1=0;m2=0;m3=0;m4=1;
   delay(30000);
   m1=0;m2=0;m3=1;m4=0;
   delay(30000);
   m1=0;m2=1;m3=0;m4=0;
   delay(30000);
   m1=1;m2=0;m3=0;m4=0;
   delay(30000);
void motor_stop() // motor stops
m1=0;m2=0;m3=0;m4=0;
void main()
while(1)
  mov clk(50);
  motor_stop();
  delay(60000);
  mov anticlk(50);
```

```
motor_stop();delay(60000);
```

• }

## Direction steps and speed of stepper motor

Program to control the direction, steps and speed of stepper motor

```
#include<reg51.h>
sbit m1=P0^0;
sbit m2=P0^1;
sbit m3=P0^2;
sbit m4=P0^3;
unsigned int a,b;
void delay(unsigned int x)
{
for(a=0;a<=x;a++);
}
void motor_clk(int y, int z) // motor moves in clock wise direction
{
for(a=0; a<=y; a++)
{
    m1=1;m2=0;m3=0;m4=0;
 delay(z);
    m1=0;m2=1;m3=0;m4=0;
    delay(z);
    m1=0;m2=0;m3=1;m4=0;
    delay(z);
    m1=0;m2=0;m3=0;m4=1;
    delay(z);
   }
void motor_anticlk(int u, int v)
                               // motor moves in anticlock wise direction
{
for(a=0; a<=u; a++)
{
m1=0;m2=0;m3=0;m4=1;
    delay(v);
   m1=0;m2=0;m3=1;m4=0;
```

```
delay(v);
   m1=0;m2=1;m3=0;m4=0;
   delay(v);
   m1=1;m2=0;m3=0;m4=0;
   delay(v);
  }
}
void motor stop() // motor stops
{
m1=0;m2=0;m3=0;m4=0;
}
void main()
while(1)
  mov_clk(50,1000);
  motor_stop();
  delay(60000);
  mov_anticlk(50,1000);
  motor_stop();
  delay(60000);
```

## HC-05 Bluetooth Module

- Wireless Serial Communication Using HC-05 Bluetooth Module
- HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.

The HC-05 Bluetooth Module can be used in a Master or Slave configuration, making it a great solution for wireless communication.

This serial port Bluetooth module is fully qualified **Bluetooth V2.0+EDR (Enhanced Data Rate)** 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. The Bluetooth module HC-05 is a MASTER/SLAVE module.

By default the factory setting is SLAVE.

The Role of the module (Master or Slave) can be configured only by AT COMMANDS.

The slave modules cannot initiate a connection to another Bluetooth device, but can accept connections.

Master module can initiate a connection to other devices.

The user can use it simply for a serial port replacement to establish connection between MCU and other device wirelessly.

#### Hardware Features

- •3.3 to 5 V I/O.
- •PIO(Programmable Input/output) control. •UART interface with programmable baud rate. •With integrated antenna.

#### Software Features

- •Slave default Baud rate: 9600, Data bits:8, Stop bit:1,Parity:No parity.
- •Auto-connect to the last device on power as default. •Permit pairing device to connect as default. •Auto-pairing PINCODE:"1234" as default.

## Pin Description

- The HC-05 Bluetooth Module has 6pins. They are as follows:
- ENABLE: When enable is pulled LOW, the module is disabled which means the module will not turn on and it fails to communicate. When enable is left open or connected to 3.3V, the module is enabled i.e the module remains on and communication also takes place.
- Vcc: Supply Voltage 3.3V to 5V
- GND: Ground pin
- TXD & RXD: These two pins acts as an UART interface for communication
- Pin Description

STATE: It acts as a status indicator.

When the module is not connected to / paired with any other Bluetooth device, signal goes Low. At this low state, the led flashes continuously which denotes that the module is not paired with other device.

When this module is connected to/paired with any other Bluetooth device, the signal goes High. At this high state, the led blinks with a constant delay say for example 2s delay which indicates that the module is paired.

**BUTTON SWITCH:** This is used to switch the module into AT command mode. To enable AT command mode, press the button switch for a second. With the help of AT commands, the user can change the parameters of this

module but only when the module is not paired with any other BT device. If the module is connected to any other Bluetooth device, it starts to communicate with that device and fails to work in AT

command mode.

### **APPLICATION**

- Wireless mobile controlled HOME AUTOMATION WIRELESS MOBILE CONTROLLEED ROBOT.
- Wireless nodes to received data to mobiles.
- Send data wirelessly from mobile to microcontroller to turn on and off the LEDs on 8051 microcontroller board.

Slave

```
#include <reg52.h>
void main()
{
unsigned char temp;
TMOD = 0x20; // timer1 mode2 SCON=0x50; // mode1 r-en
TH1=0x0FD; //9600 baud rate TR1=1; //on timer1
if (temp=='1')
P1=0x00; // port 1 off
while(1)
```

```
{
P1=(0X01<<4); // LED1 glow
}
else if (temp=='2')
P1=(0X01<<5); // LED1 glow
else if (temp=='3')
Logic to generate control action
{
P1=(0X01<<6); // LED1 glow
based on received data }
//rx while(!RI); // wait till RI become 1
Wait till serial data is received
temp=SBUF; // collect received data into variable temp RI=0; // Clear RI flag else if (temp=='4')
{
P1=(0X01<<7); // LED1 glow }
else if (SBUF=='5')
P1=(0X00); // LED1 off }
ESP - 8266 Wifi module
```

• The ESP8266 can be controlled from your local Wi-Fi network or from the internet (after port forwarding). The ESP-01 module has GPIO pins that can be programmed to turn an LED or a relay ON/OFF through the internet. The module can be programmed using an Arduino/USB-to-TTL converter through the serial pins (RX,TX).

Communication With Nodemcu UART I2C SPI

- UART, or Universal Asynchronous Receiver/Transmitter, is a form of serial communication that relies on just one wire going in either direction. UART is done via pins Rx and Tx, which are respectively used to receive and transmit.
- Inter-integrated circuit does have a separate clock signal, but uses just one wire for data transmission. It's great for connecting a single master device to multiple slaves, each of which has a separate address.
- serial peripheral interface, It's commonly used to connect microcontrollers and other integrated circuits. It's also full-duplex, which means that every read operation is able to coincide with a write operation. On the NodeMCU, SPI uses three pins: D5 is the CLK; D6 is the Master In Slave Out (or MISO); D7 is the Master Out Slave In (MOSI).

## Sensors with Arduino

fritzing

```
GND
                                  GND
          HC-SR04
          DHT 11
          IR Sensor
Program for HC-SR04
#define echopin 7
#define trigPin 7
Long duration, distance:
Void setup()
{
Serial.begin(9600);
PinMode(trigpin,OUTPUT);//Trigger pin
PinMode(echopin,INPUT);//Echo pin
}
Void loop()
{
digitalWrite (trigPin,LOW);
DelayMicroseconds(2);
DigitalWrite(trigPin,LOW);
//Now lets read the bounced wave
duration = pulseln(echoPin, HIGH);
//calculate the distance
distance = (duration/58.138)*.39;
Serial.print("cm");
Serial.print(distance);
Delay(1000);
```

## Serial communication

- Parallel
- Serial
- UART in 8051

- An UART, universal asynchronous receiver / transmitter is responsible for
- performing the main task in serial communications with computers. The device changes incoming parallel information to serial data which can be sent on a communication line. A second **UART** can be used to receive the information.
- To communicate with pc or other devices we need some standards like RS232.
- A driver(MAX232) is used to convert TTL level to RS232 voltage level. **PROJECTS**

#### FIRST ROBOT

Device used -

Microchip, AT89C51

Program to be used

```
//PROGRAM TO FOLLOW
#include<reg51.h>
sbit mot1p=P1^0;
sbit mot1n=P1^1;
sbit mot2p=P1^2;
sbit mot2n=P1^3;
//connectivity_of_the_motors//
sbit sw1=P2^0;//forward direction
sbit sw2=P2^1;//reverse direction
sbit sw3=P2^2;//left_direction
sbit sw4=P2^3;//right_direction
sbit sw5=P2^4;//stop
void main()
   mot1p=mot1n=mot2p=mot2n=0;
   while(1)
     if(sw1==0)
        mot1p=1;
        mot1n=0;
        mot2p=1;
        mot2n=0;//robot moves forward direction
     }
      if(sw2==0)
        mot1p=0;
        mot1n=1;
        mot2p=0;
        mot2n=1;//robot moves reverse direction
     }
if(sw3==0)
        mot1p=0;
        mot1n=1;
        mot2p=1;
        mot2n=0;//robot moves left direction
     }
      if(sw4==0)
        mot1p=1;
```

```
mot1n=0;
          mot2p=0;
          mot2n=1;//robot_moves_right_direction
         if(sw5==0)
         mot1p=mot1n=mot2p=mot2n=0;
  • }
OUTPUT
M
     LCD
     Device used - LM016L, AT89C51
//PROGRAM_TO_FOLLOW
send_data('D');
OUTPUT
<u>Timer</u><u>Device used - AT89C51</u>
//PROGRAM_TO_FOLLOW
OUTPUT
PWM
  • Device used - AT89C51
//PROGRAM_TO_FOLLOW
OUTPUT
```

) mStm ws V 5 mV V 5 mV