Microcontroller Based System Design & Development

Subject Code: 28564

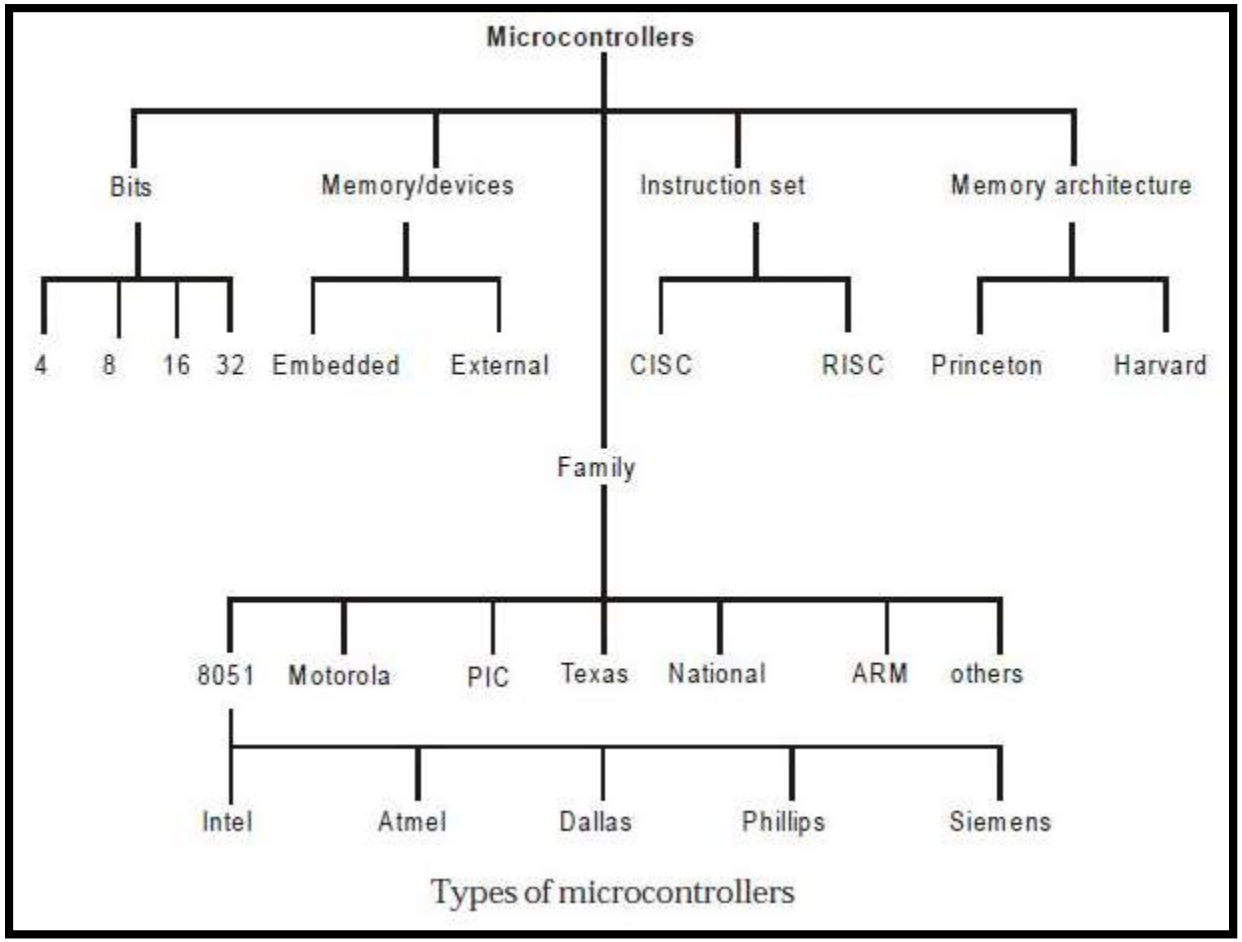
Presented By **Obydul Islam**
Junior Instructor (Part-time),
Computer Science & Technology, MPI

Chapter-1

- Microcontroller:

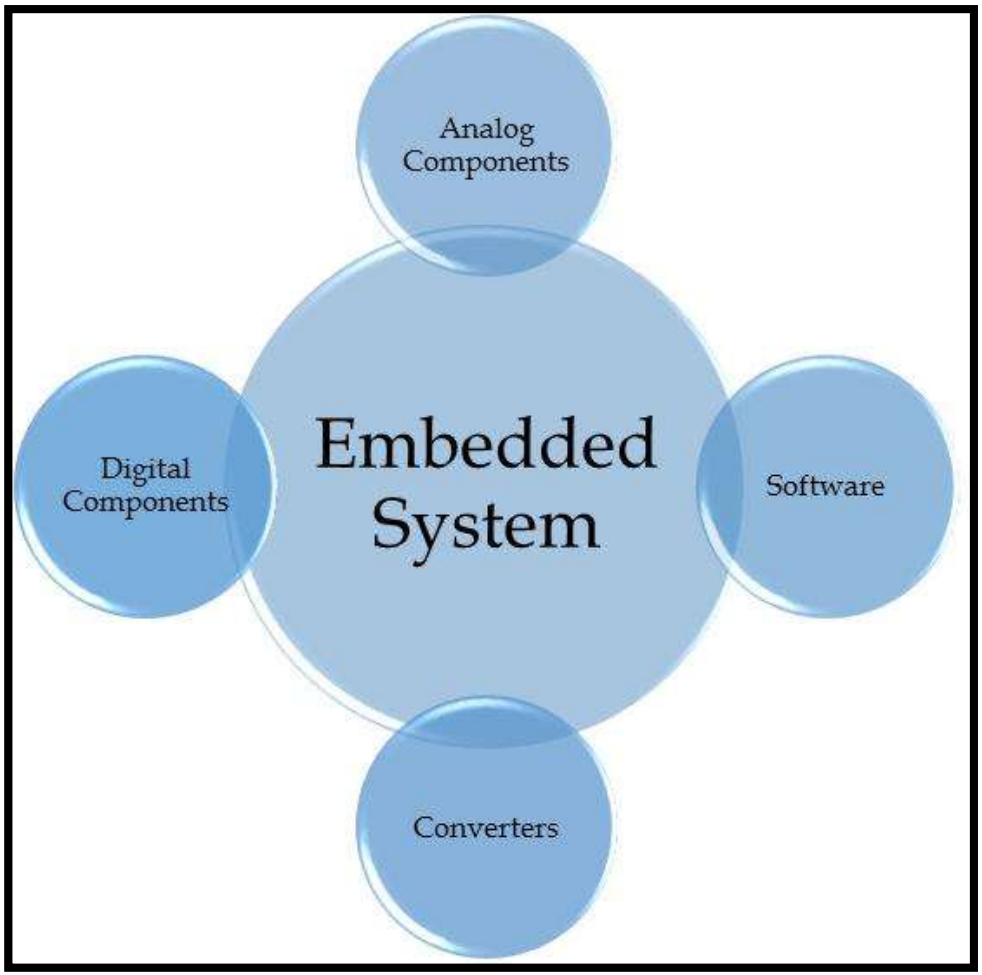
- In a microcomputer system, the device created by integrating chips or devices, data storage, and input/output ports with the microprocessor used for control work, to communicate with the external world and receive signals, along with memory input/output interfaces and other auxiliary peripherals, is called a microcontroller.

Different types of microcontroller:



Types of microcontrollers

Embedded System:

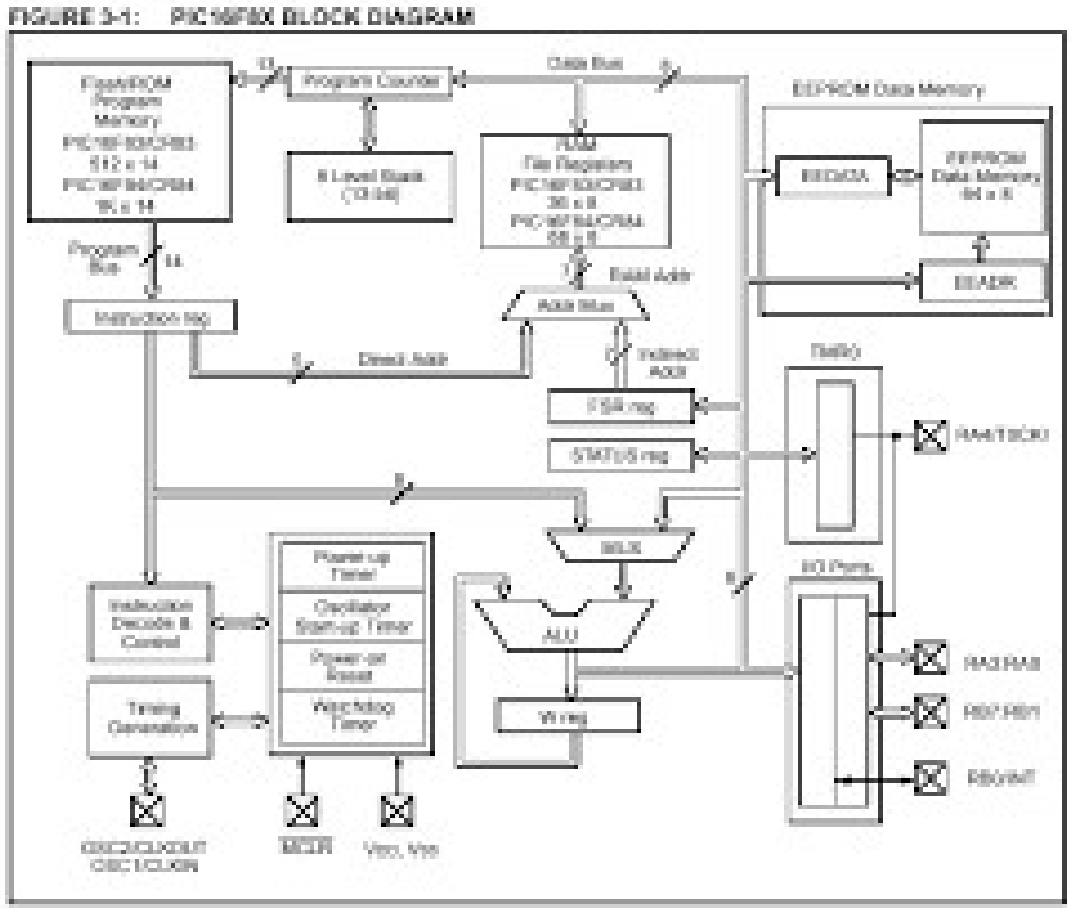


Chapter-2

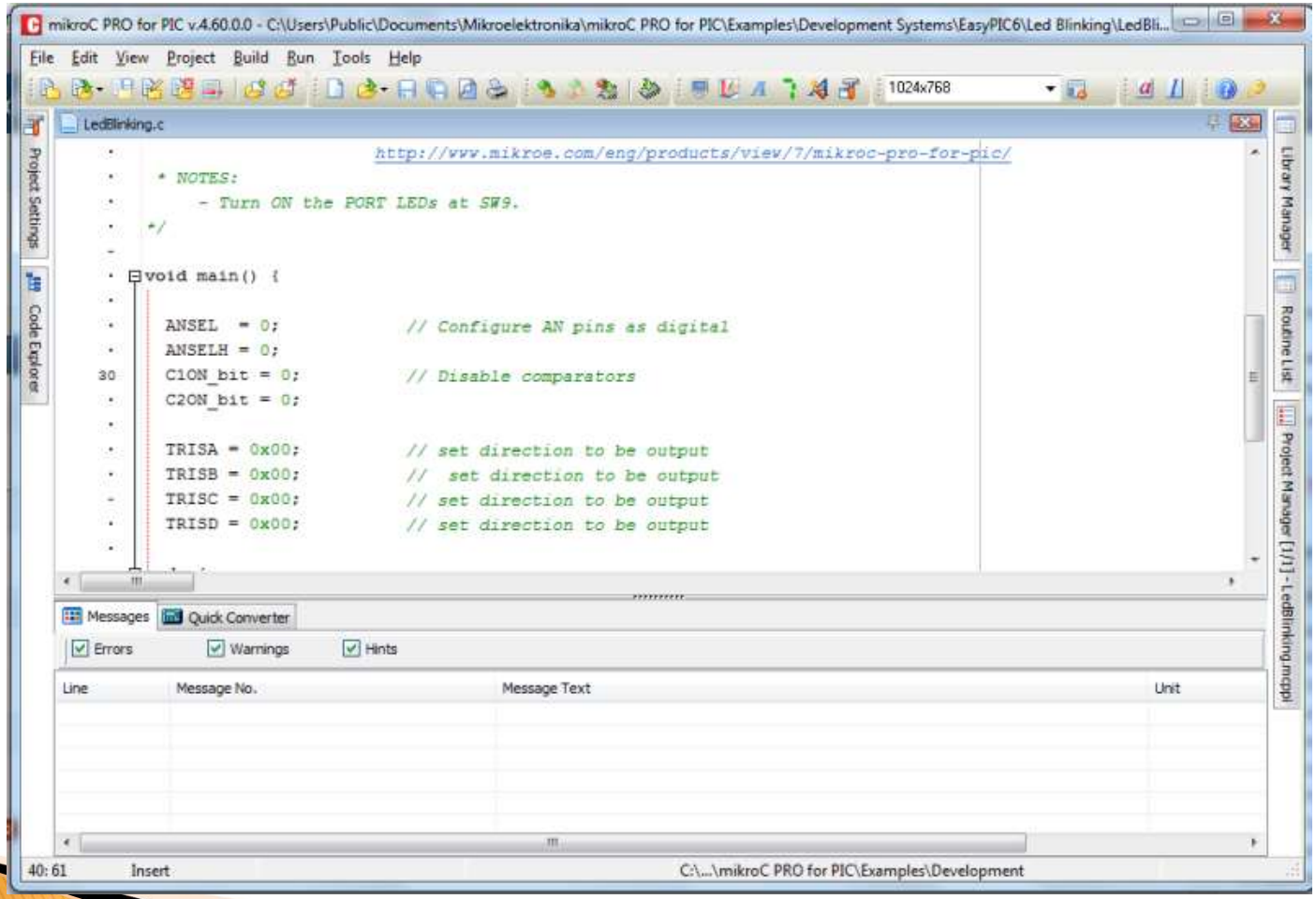
- PIC microcontroller:

- PIC is a Peripheral Interface Microcontroller which was developed in the year 1993 by the General Instruments Microcontrollers. It is controlled by software and programmed in such a way that it performs different tasks and controls a generation line. PIC microcontrollers are used in different new applications such as smartphones, audio accessories, and advanced medical devices.

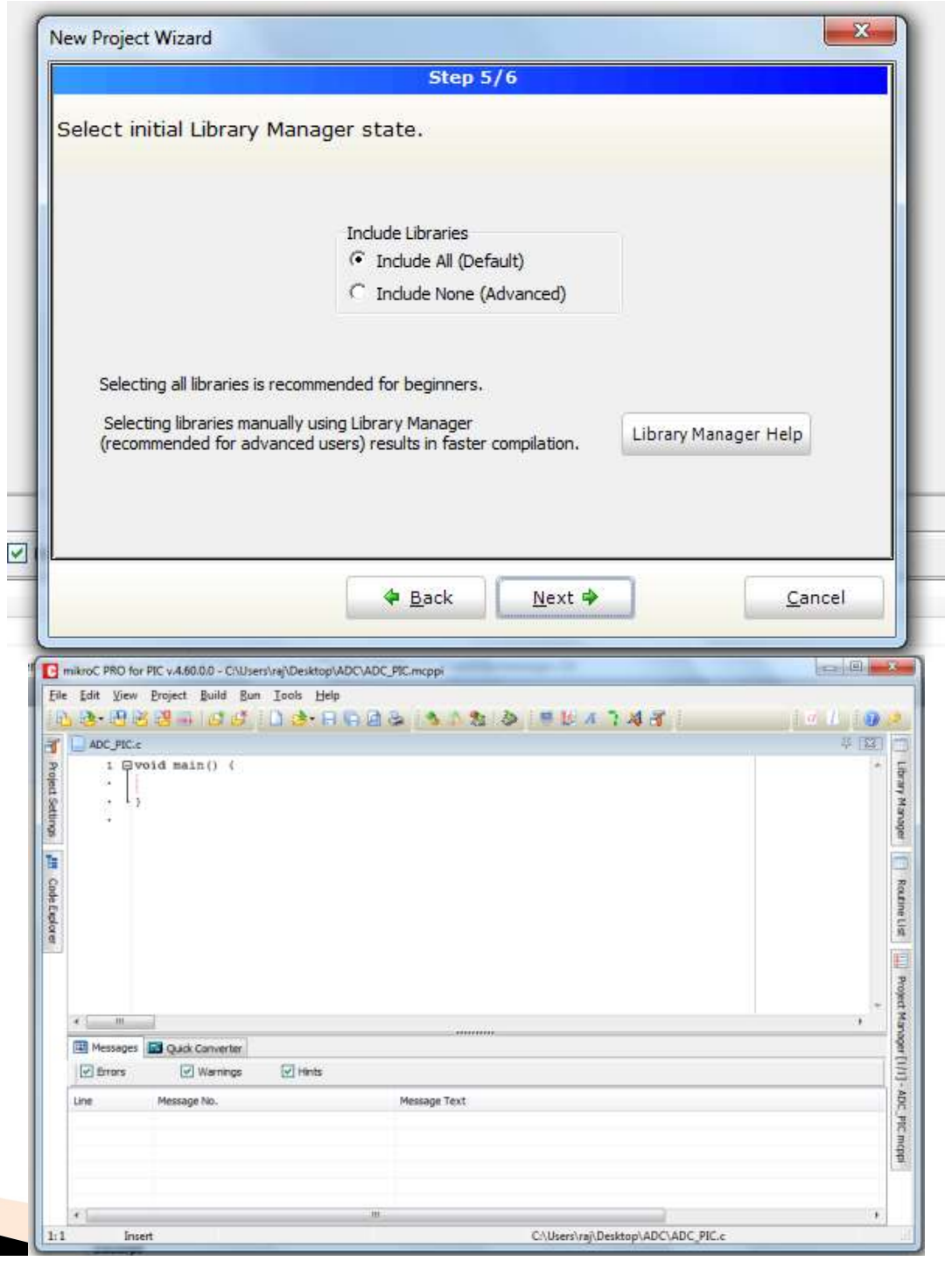
Mid Range Microcontroller



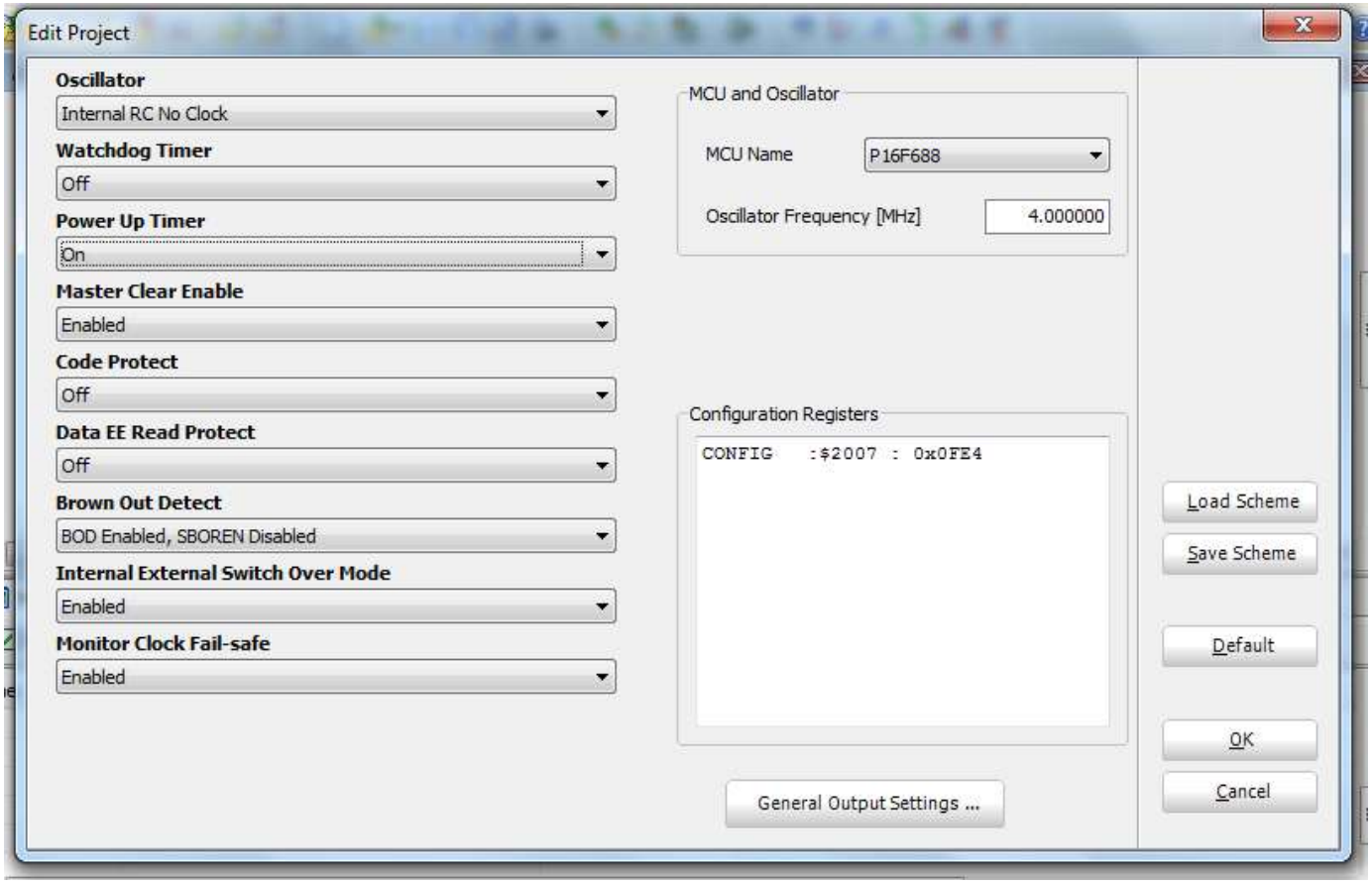
Chapter-3
MikroC Compiler Installation



My PIC tutorials and projects use MikroC compiler for firmware development. But I don't think I ever posted anything on its installation and setup. Today, I am going to show how to install MikroC Pro for PIC (v4.60) on a Windows PC. First of all, download the zipped installation file from here, unzip it and run the setup program.



Installation is straightforward. When you first start the MikroC compiler, it opens a LED blinking example project. You can close this project by clicking on 'Close Project' under Project menu.



Chapter–4
Timer/ Counter Program

- package main
- import "fmt“
- func (x *Timer) tick(){
- x.id++ fmt.Println(x.id) }
- type Timer struct {
- value string id int }
- func main() {
- var x int fmt.Scanln(&x) t := Timer{"timer1", 0}
- for i:=0;i<x;i++ { t.tick() } }

Chapter-5
Microcontroller Interrupt

UART Example for PIC16F887 CCS C code:

The code used in this example is shown below.

The function #use rs232(UART1, baud = 9600) is used to configure the UART protocol. Here the hardware UART module is used. If the pins TX and RX (RC6 and RC7) are used by an other application we can use software UART. Software UART is generated by the compiler with the same previous function. For example TX is mapped to pin RD0 and RX to pin RD1:

#use rs232(xmit = PIN_D0, rcv = PIN_D0, baud = 9600)

Hardware Int.

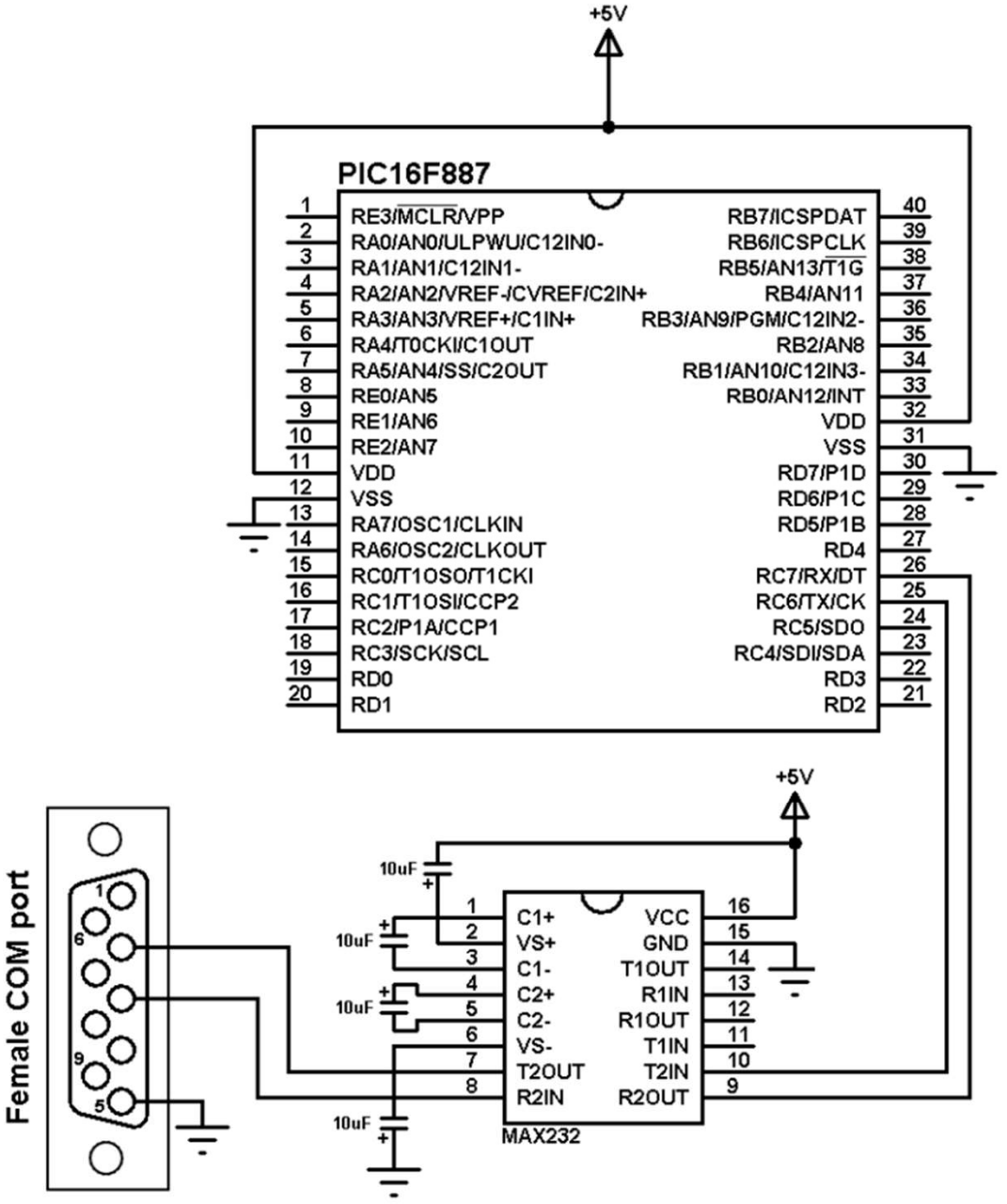
The functions used in the C code are:

printf: sends a string of characters over RS232 transmission pin (TX).

putc: send a character over RS232 transmission pin (TX).

if(kbhit()): test if a character is ready for getc() function.

getc(): read the character.



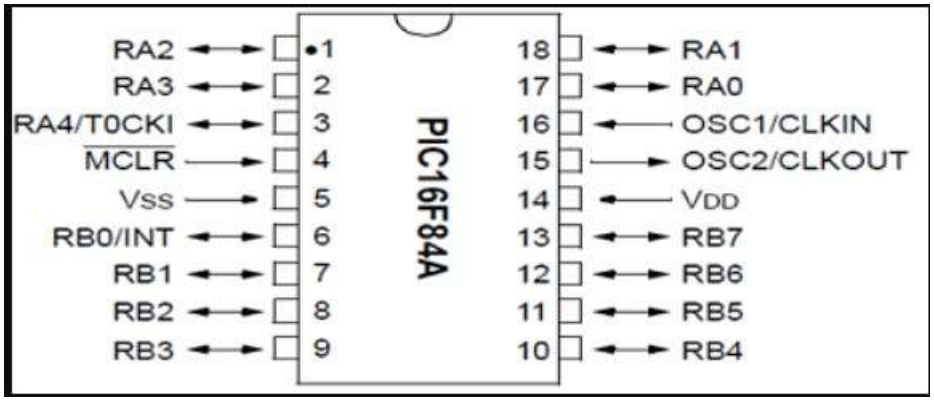
CCSS C Code Example

```
// CCS C UART example for PIC16F887 microcontroller

#include <16F887.h>
#fuses NOMCLR, INTRC_IO, NOBROWNOUT, NOLVP
#use delay(clock = 8MHz)
#use rs232(UART1, baud = 9600)

const char message[] = "PIC16F887 microcontroller UART example" ;
int8 i = 0, j;
void main(){
  setup_oscillator(OSC_8MHZ);              // Set internal oscillator to 8MHz
  delay_ms(2000);                          // Wait 2 seconds
  // Print text
  printf("Hello world!");
  // Print list of characters
  printf("\n\r");                          // Start new line
  while(message[i] != '\0'){
```

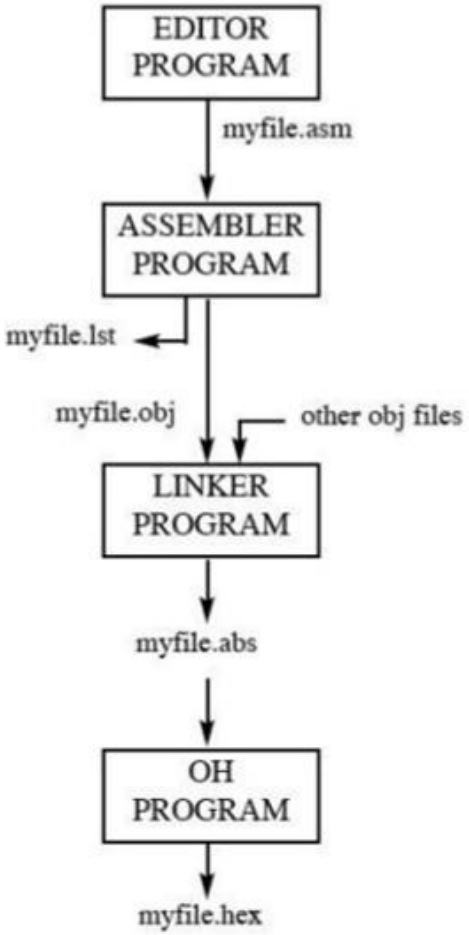
```
    putc(message[i]);                    // Write character
      delay_ms(100);                     // Wait 100 ms
      i++;                               // Increment i
    }
    // Print numbers
    printf("\n\r");                      // Start new line
    for(i = 0; i < 21; i++){
      printf("%u\n\r", i);               // Print i and start new line
      delay_ms(500);
    }
    // Receive and send data via UART
    while(TRUE){
      if(kbhit()){                       // If a character available
        j = getc();                      // UART read
        putc(j);                         // Send it back
      }
    }
  }
```

Chapter-6

Mid Range PIC Microcontroller Pin Diagram

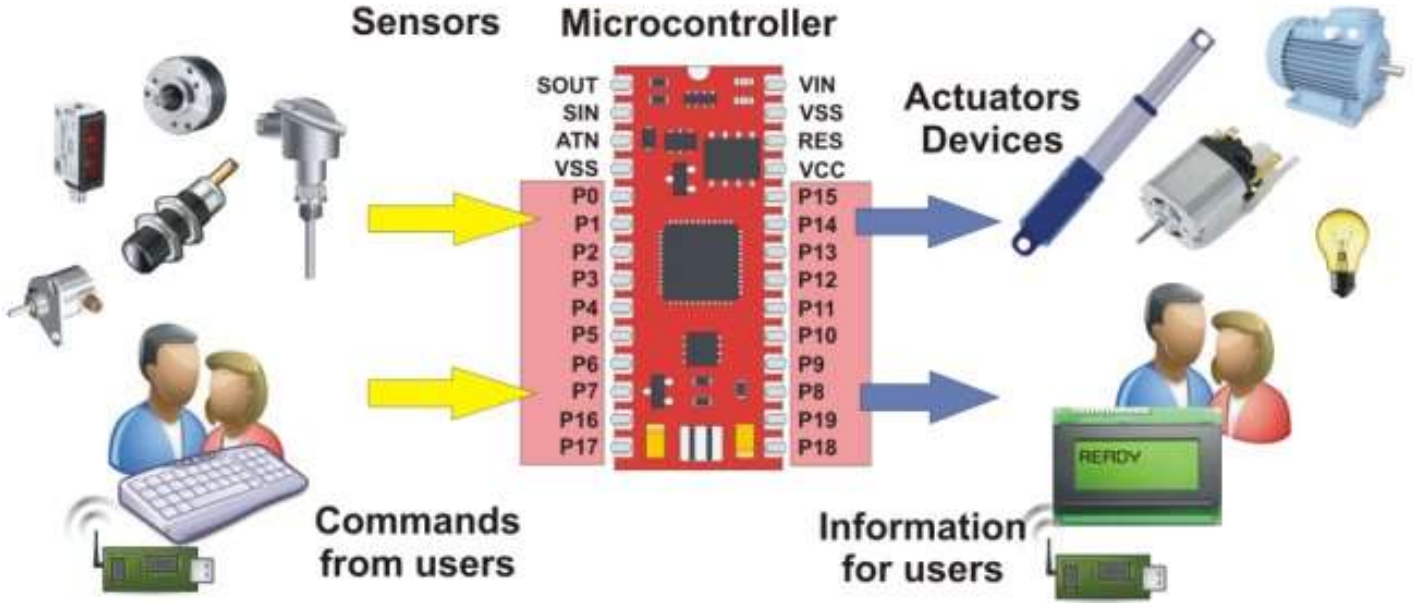


Step to create and execute assembly language program



Chapter-7

Interfacing process of sensors and actuators in an embedded system

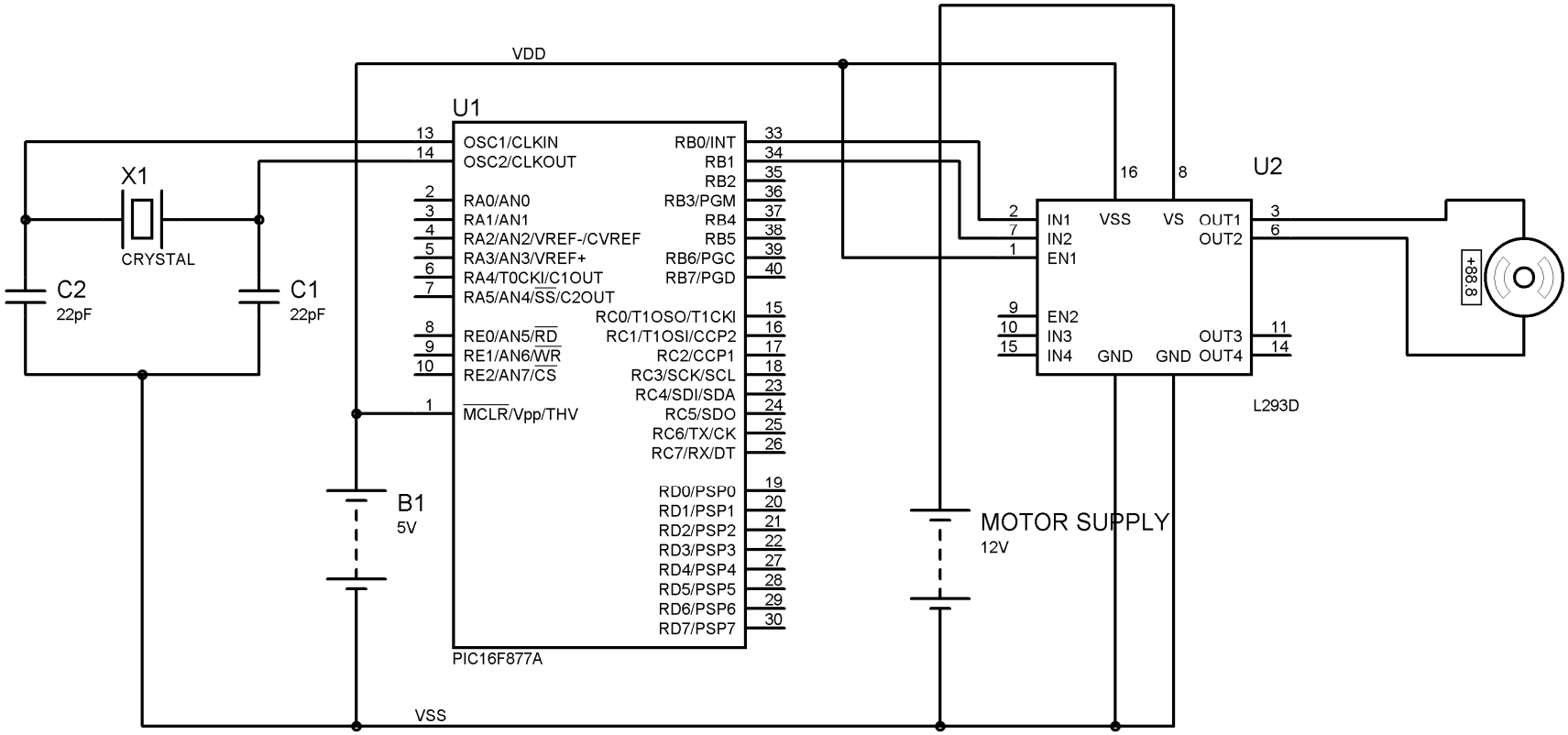


- **Understanding Sensors:**
- Sensors are devices that detect and measure physical phenomena such as temperature, pressure, light, motion, and more. They act as the eyes and ears of an embedded system, providing valuable data for analysis and decision-making. Common types of sensors include:
- **Temperature sensors:** Monitor changes in temperature, vital for applications like climate control systems or industrial processes.
- **Pressure sensors:** Measure pressure variations, essential in automotive applications, weather monitoring, and medical devices.
- **Proximity sensors:** Detect the presence or absence of objects, commonly used in automation, robotics, and touch-sensitive interfaces.
- **Accelerometer:** Measure acceleration and tilt, found in devices like smartphones, gaming consoles, and drones.
- **Actuators:** Bringing Actions to Life:
- Actuators are components that convert electrical signals from the embedded system into physical actions. They enable the system to manipulate or control various devices and mechanisms. Some commonly used actuators include:
- **Motors:** Convert electrical energy into mechanical motion, utilized in robotics, industrial automation, and automotive systems.

Integration of Sensors and Actuators:

- The true power of sensors and actuators lies in their seamless integration within embedded systems. Micro controllers or microprocessors serve as the brain of the system, orchestrating the interaction between sensors, actuators, and other peripherals. The steps involved in this integration include:
- **Sensor data acquisition:** Analog signals from sensors are converted into digital data using analog-to-digital converters (ADCs).
- **Signal processing:** The acquired data is analyzed and processed using algorithms to extract meaningful information.
- **Actuator control:** Based on the processed data, appropriate signals are generated to control the actuators and trigger desired actions.
- **Applications of Sensors and Actuators in Embedded Systems:**
- Sensors and actuators find extensive applications in various domains, including:

A Program for controlling motors and LEDs using microcontrollers

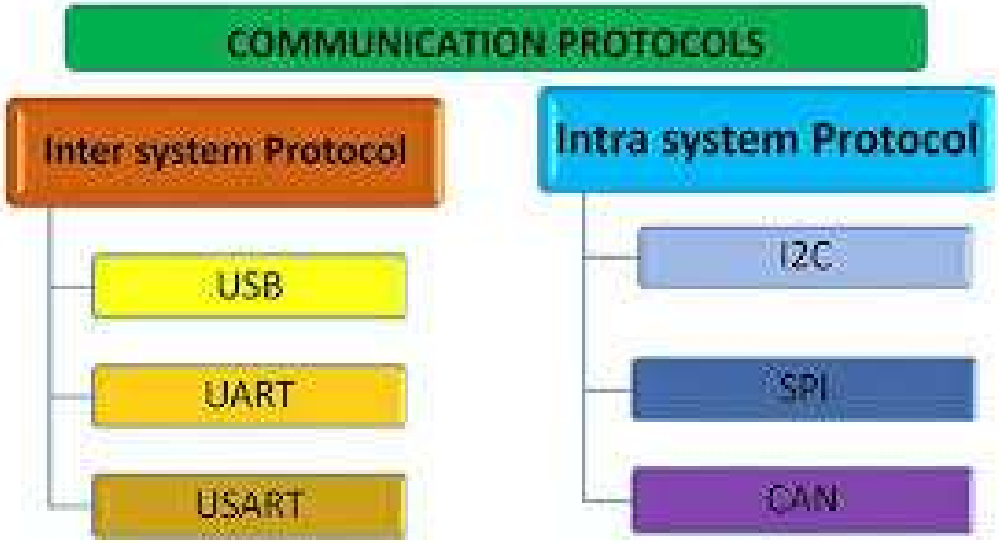


```
#include <stdio.h>
#include <stdlib.h>
 #include <includes.h>
 void main() { // Setting up PIC modules such as Timers, IOs OCs,Interrupts, ...
InitializeIO();
InitializeLEDs(); InitializeTimers();
while(1) { WaitOnBtn1();
Forward(4.0,70); Stop(1.0);
Backward(3.0,50); Stop(2);
 Forward(3.0,40); Stop(1.0);
Backward(2.0,20); LEDsOFF(); } return; }
void InitializeIO()
{ TRISAbits.TRISA6 = 1;
TRISAbits.TRISA7 = 1;
 TRISGbits.TRISG12 = 0;
 TRISGbits.TRISB13 = 0;
 LATGbits.LATB12 = 0;
LATGbits.LATB13 = 0; return; }
void InitializeLEDs(){
 //code to initialize LEDS }
void InitializeTimers(){
// Initialize Timer1 T1CON = 0x0000;
 // Set Timer1 Control to zeros T1CONbits.TCKPS=3;
```

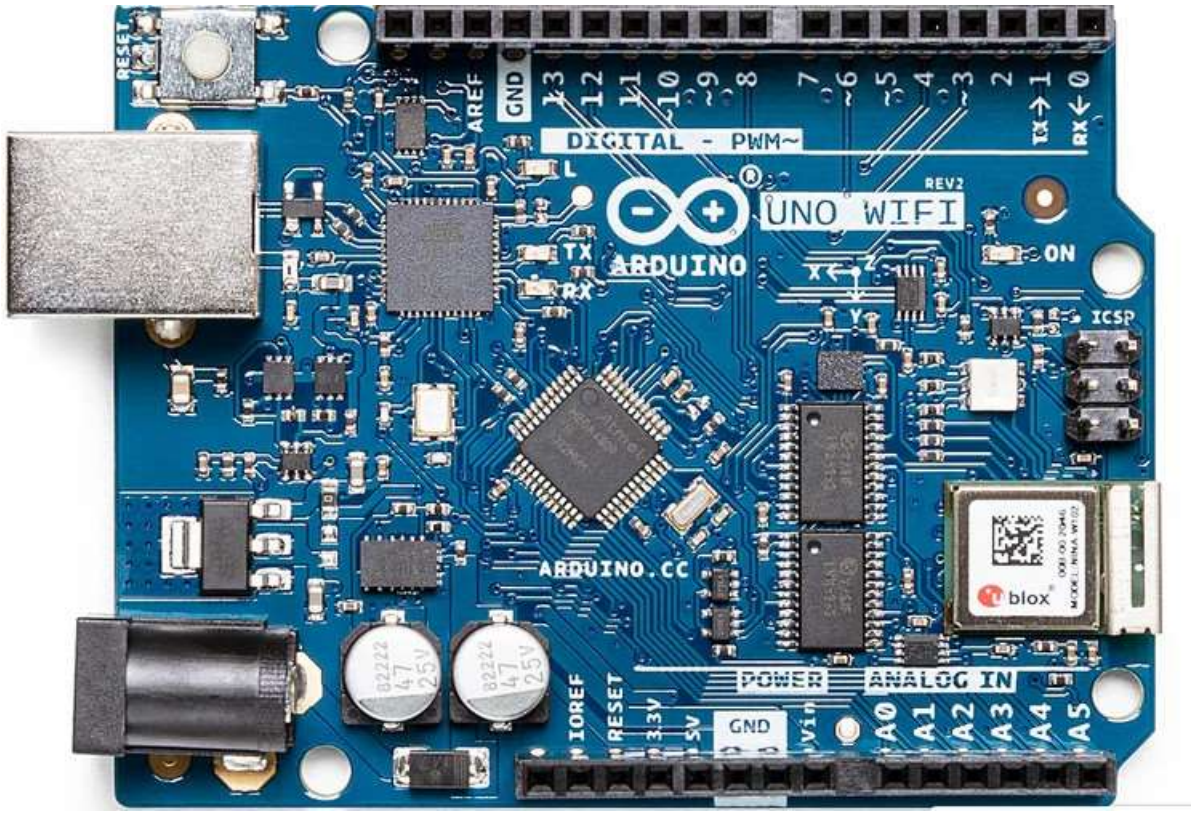
```
 // prescale by 256 T1CONbits.ON = 1;
 // Turn on Timer PR1= 0xFFFF;
// Period of Timer1 to be full TMR1 = 0;
 // Initialize Timer1 to zero
 // Initialize Timer2 T2CON = 0; T2CONbits.TCKPS = 7;
 // prescale by 256 T2CONbits.T32 = 1;
 // use 32 bits timer T2CONbits.ON = 1;
PR2 = 0xFFFFFFFF; // Period is set for 32 bits TMR2 = 0; }
 void WaitOnBtn1(){
 // wait on Btn1 indefinitely while(PORTAbits.RA6 == 0);
// Turn On LED1 indicating it is Btn1 is Pushed LATBbits.LATB10 = 1;
 return; }
void Forward(float Sec, int D){
int RunTime = (int)(Sec*39000);
// convert the total time to number of Tics TMR2 = 0; //LEDs LATGbits.LATG12
= 1;
// forward Direction LATBbits.LATB12 = 0; LATBbits.LATB13 = 0;
LATBbits.LATB11 = 1;
 // Keep on firing the PWM as long as Run time is not elapsed while (TMR2 <
RunTime){ PWM(D); }
 return; }
void PWM(int D){ TMR1 = 0; int Period = 400; while (TMR1< Period) { if (TMR1
```

Chapter-8

Communication Protocol in Embeded System



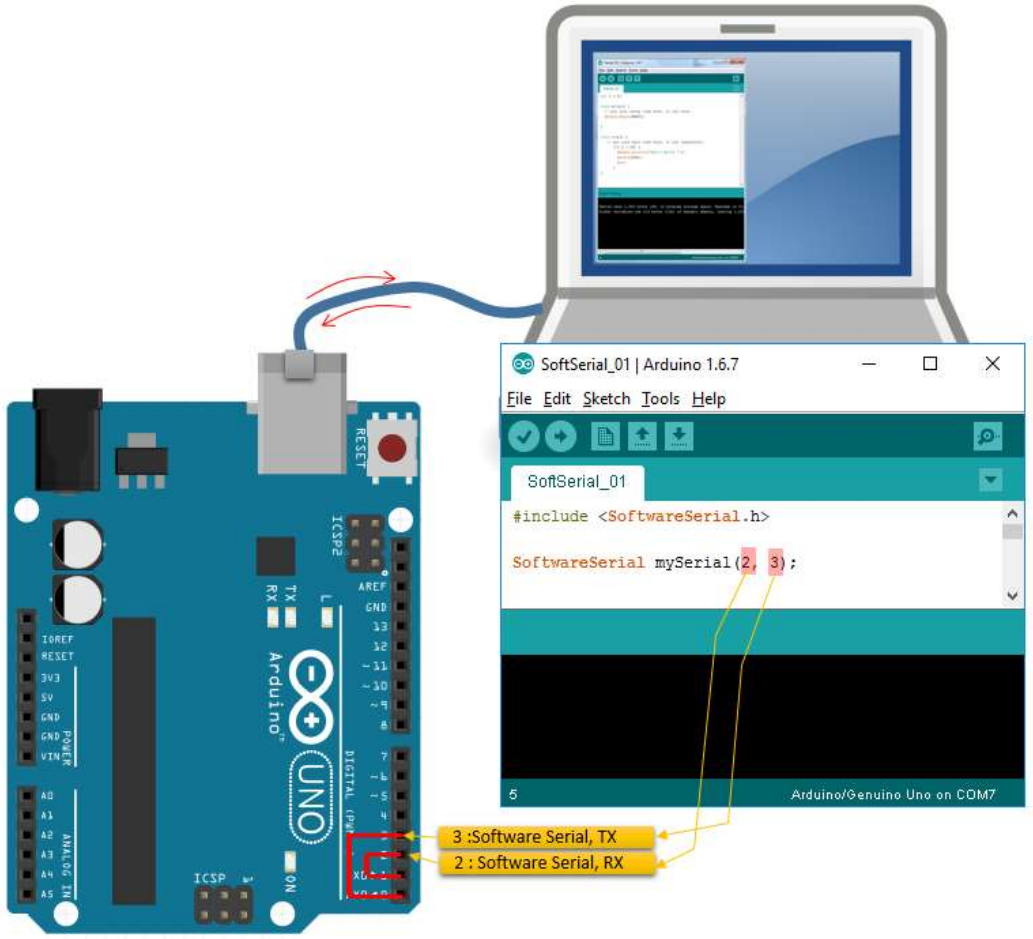
Chapter- 9
Arduino



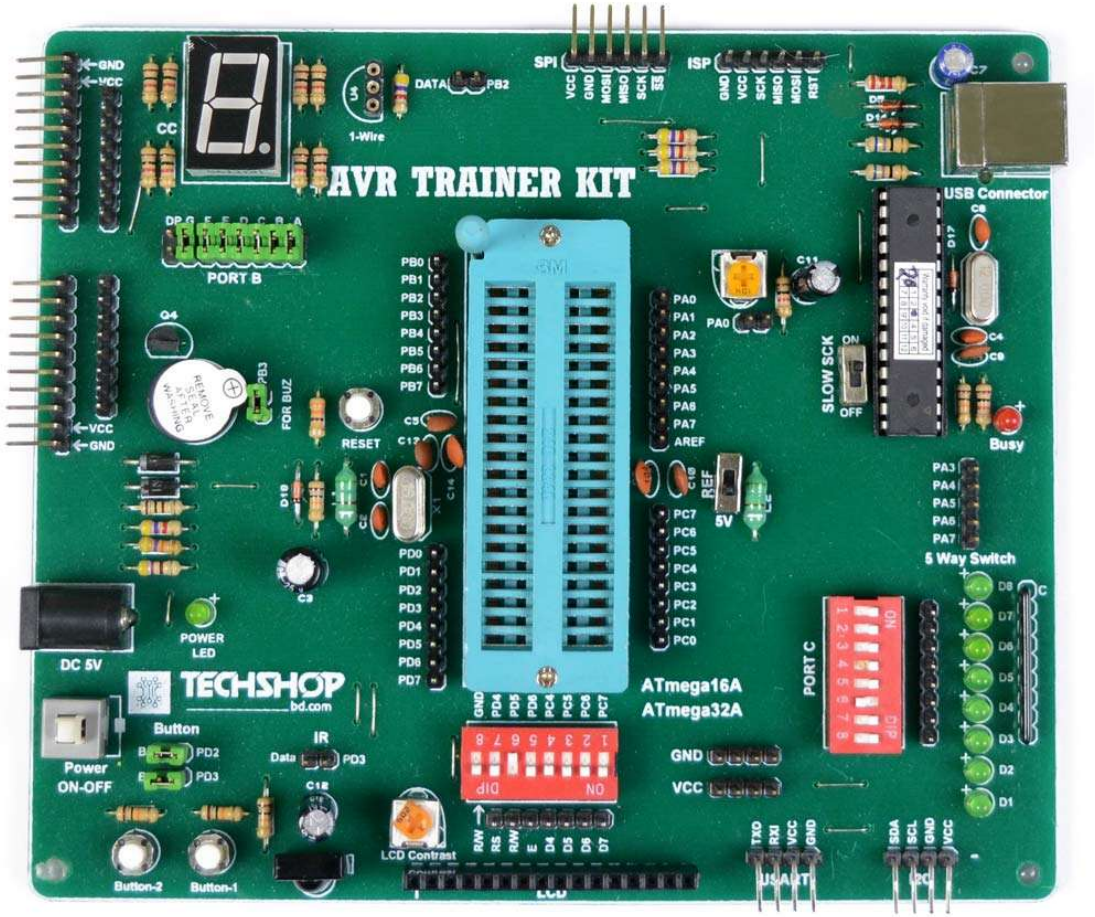
- **Arduino** is an Italian open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Its hardware products are licensed under a CC BY-SA license, while the software is licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL),[1] permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially from the official website or through authorized distributors.
- The Arduino project began in 2005 as a tool for students at the Interaction Design Institute Ivrea, Italy,[3] aiming to provide a low-cost and easy way for novices

Connect Aurdion With pc and communicate

You can use the Arduino environment's built-in serial monitor to communicate with an Arduino board. Click the serial monitor button in the toolbar and select the same baud rate used in the call to begin() . Serial communication on pins TX/RX uses TTL logic levels (5V or 3.3V depending on the board).



Chapter-10
Different Microcontroller Based Development Kit



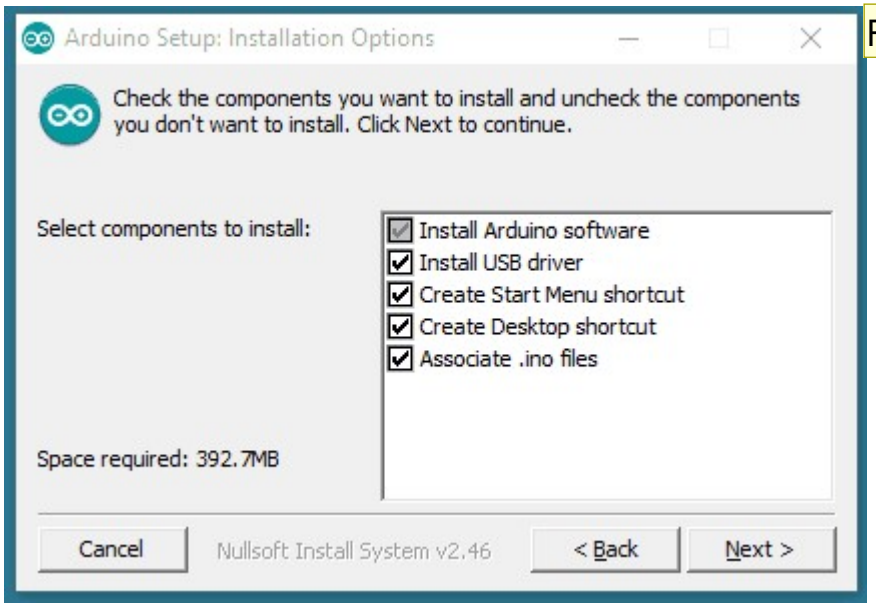
AVR Trainer KIT PRO


AT89S52


HL-K18
HL-K18 PIC
ROBOTICSBD
CR1220

Download the Arduino Software (IDE)

Get the latest version from the **download page**. You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE),



F1

F1 FUYAD, 8/6/2024

including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create a **portable installation**.
When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.