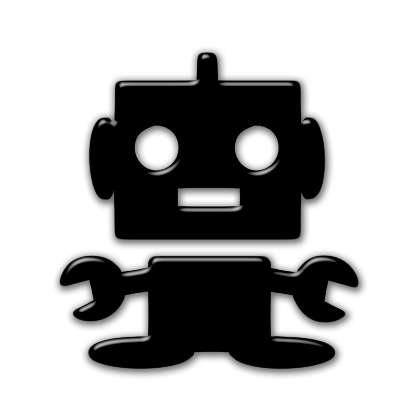
**[](https://www.google.co.in/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUKEwj6y7_6n6fLAhUNC44KHSy9AZUQjRwIBw&url=http://icons.mysitemyway.com/legacy-icon-tags/robot/&bvm=bv.115339255,d.c2E&psig=AFQjCNGhTwDgMHk60biQEjSJWs-_ZlG6qw&ust=1457188388774577)**

**INDEX**

|  |  |  |
| --- | --- | --- |
| S.No: | TOPIC NAME: | PAGE No: |
| I | **PREFACE** | **3-5** |
| II. | **ABSTRACT** | **6-7** |
| III. | **LIST OF COMPONENTS** | **8-10** |
| IV. | **Block Diagram** | **11-13** |
| v | **Circuit Diagram** | **14-16** |
| VI | **Circuit Description** | **17-20** |
| VII | **Working** | **21-22** |
| VIII | **Hardware Description** | **23-51** |
| IX | **SOFTWARE DISCRIPTION** | **52-71** |
| X | **RESULT** | **72-74** |

**In many application of controlling robotic gadget it becomes quite hard and complicated when there comes the part of controlling it with remote or many different switches.**

**Mostly in military application, industrial robotics, construction vehicles in civil side, medical application for surgery. In this field it is quite complicated to control the robot or particular machine with remote or switches, sometime the operator may get confused in the switches and button itself, so a new concept is introduced to control the machine with the movement of hand which will simultaneously control the movement of robot.**

**We use ultrasonic sensor to measure the distance of an object from the required station. The main purpose of the sensor is to generate high frequency waves and get the information in the form of feedback waves. This feedback waves gives the information about object at what distance it is.**

**In this we use a Bluetooth module; it encodes the signals/information from the ultrasonic to the control station. Here we use a special device, Camera which is an optional one. It is specially used for live streaming, sometimes for recording also.**

* **Applications:**
* **RAW agents**
* **Archeologists**
* **Airways**
* **Astronauts**

**[](https://www.google.co.in/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUKEwj1zfGfrafLAhVQCY4KHUi1BQQQjRwIBw&url=http://www.dreamstime.com/photos-images/detective.html&psig=AFQjCNGqZi7mhLa2X6DPYs4nctGMtK66Tw&ust=1457191964344106) [](https://www.google.co.in/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=&url=http://www.buzzle.com/articles/famous-archaeologists.html&psig=AFQjCNFe5Odau8Sp9tjR3ynoBOxo_4Sr2w&ust=1457192010300687)**

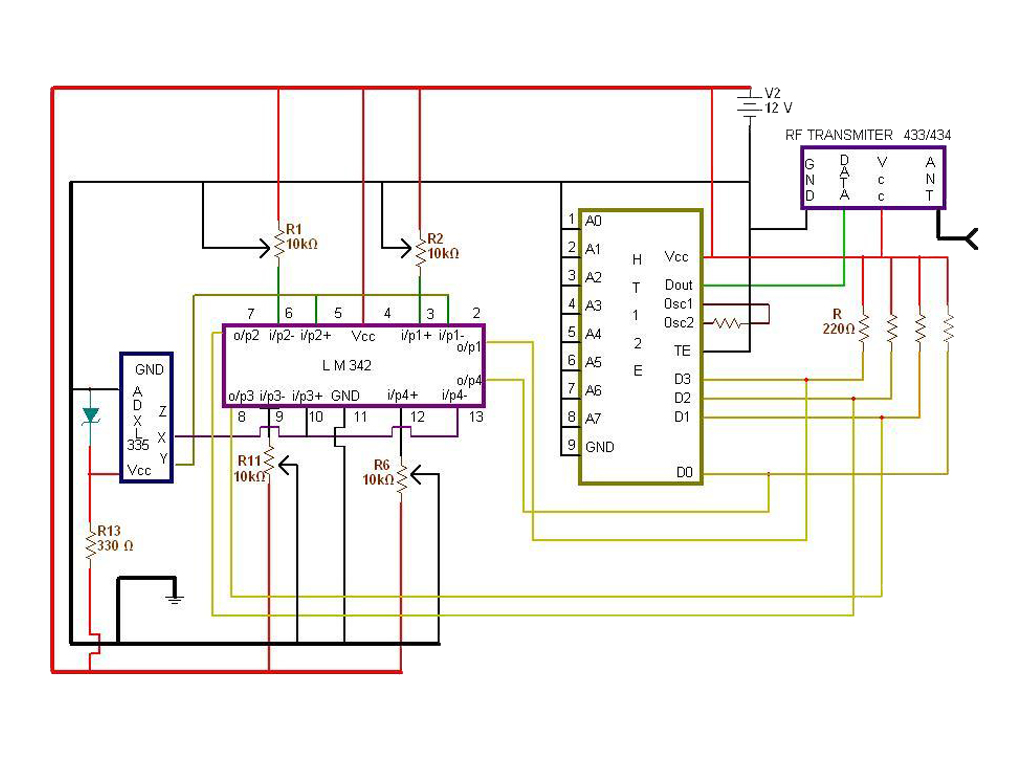
**[](https://www.google.co.in/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUKEwi2tMjtrafLAhXUkY4KHc8OAgIQjRwIBw&url=http://www.holidayextras.co.uk/flight/tg911.html&psig=AFQjCNGbfhDkJFDG5x1ADVw3b9gpIT9-mQ&ust=1457192095280710) [](https://www.google.co.in/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUKEwjynYiJrqfLAhUNcY4KHak4DksQjRwIBw&url=http://www.123rf.com/photo_35068646_stock-vector-black-and-white-vector-illustration-of-an-astronaut-in-a-space-suit.html&psig=AFQjCNHHLJDPOBRS1Z9EQhiLZg7UsxEVEg&ust=1457192164664335)**

* **Though robots have certain limitations in the way they are made, humans are able to utilize them perfectly based on the desired applications. Robotics has come a long way and new findings or developments never end. Here are some applications where robots play important roles.**
* **What robots can do?**
* **Robotics is highly advanced technology that strives to develop robots for various applications.  Let us have a look at robots already invented and being used in the industry.**
* **1) Industrial robots:**
* **2) Aerospace robots:**
* **3) Healthcare delivery:**
* **4) Robots resembling human beings and robotic pets:**
* **5) Military robots:**
* **Robots are playing an important role in automation across all the sectors like construction, military, medical, manufacturing, etc. After making some basic robots like line follower robot, computer controlled robot, etc; we have developed this accelerometer based gesture controlled robot by using microcontroller. In this project we have used hand motion to drive the robot. For this purpose we have used accelerometer which works on acceleration.**
* **Semiconductors:**
* **89s52 IC = 1**
* **LM324 IC = 1**
* **HT12E IC = 1**
* **HT12D IC = 1**
* **L293D IC = 1**
* **7805 IC = 1**
* **LED = 1**
* **ZENER DIODE 3.3 = 1**
* **Capacitors:**
* **33pf = 1**
* **104 = 1**
* **10uf = 2**
* **Resistors:**
* **10KΩ = 6**
* **120Ω = 4**
* **1KΩ =1**
* **51KΩ =1**
* **10K Variable =4**
* **Other components**
* **RF Transmitter & Receiver Module 433/434 MHz:**
* **ADXL335:**
* **HL-05 Bluetooth Module:**
* **11.0592 Crystal Oscillator:**
* **Reset Button:**
* **Female & Male Pin Connectors:**
* **HC-SR04 Ultra Sonic Sensor:**
* **Gear Motors:**
* **Wireless Camera with Module:**
* **12v Rechargeable Battery:**
* **9v Battery:**

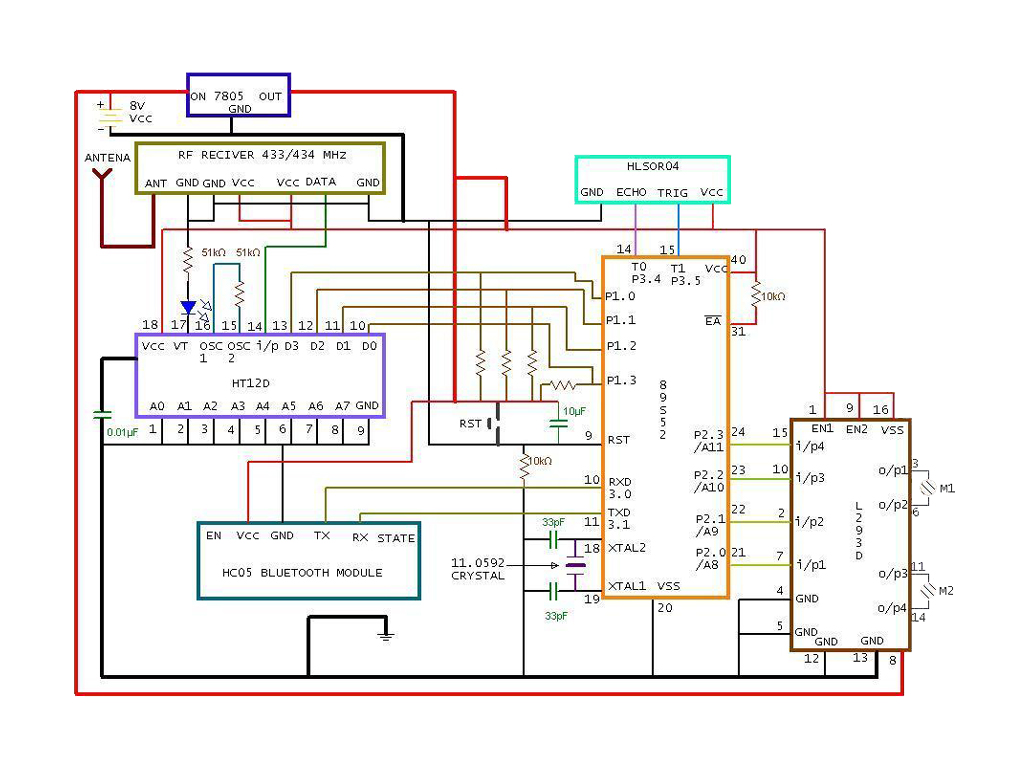
**TRANSMITTER**

**RECEIVER**

**TRANSMITTER**

****

**RECEIVER**

****

**“ADXL335” it consists of 5 pins, namely x, y, z, vcc & ground. The vcc pin is connected to power supply (5v) with 330Ω and ground terminal is connected to 3.3 volts Zener diode. It sends the axis information to “LM324” which transfers the signal from “ADXL335” and maintains sensitivity. The x-axis is connected to 2nd &5th pin of “LM324” as input. The y-axis is connected to 13th & 10th pin of “LM324” and z-axis is empty.**

**“LM324” it consists of 14 pins, among them 11th pin is used as ground and 4th is taken as vcc. The pins 3, 6, 12&9 pins are given to 10K variable resistance. “LM324” transmit the signal to “HT12E”. Pin 1, 7, 8, and 14 are output pins of “LM324” which are given as inputs to “HT12E”.**

**“HT12E” it consists of 18 pins, out of this 1 to 9 pins are grounded. The output of “LM324” is given to 10th&13th pins of“HT12E” as input. 18th pin is taken as vcc which is connected to ground and also to 10, 11, 12&13 pins with resistance 220Ω each. 15th pin is connected to 16th pin with resistor. Here “HT12E” encodes the information and it sends to the “RF-TRANSMITTER” for further operation. The 17th is connected to data pin of “RF-TRANSMITTER” (433/434 MHz).**

**“RF-TRANSMITTER” it consists of 4 pins, namely ground, data, vcc & antenna. It passes the information in the form of frequency (433/434 MHz) to the antenna. The ground pin is connected to negative of power supply and antenna transmits the signals to “RF-RECEIVER”.**

**“RF-RECEIVER” it consists of 8 pins out of this three are ground, two are vcc, and two are data and an antenna. The transmitted signals are received by the receiver antenna. It transforms the information from frequency to voltage and then it transmits the information to “HT12D”. The two vcc ­pins are connected each other. The three ground pins are connected each other and then to negative of the LED.**

**“HT12D” it consists of 18 pins. The data pin is connected to 14th pin that is input of the “HT12D”. 15th&16th pins (OSC1&OSC2) are connected each other with 51K resistor. The LED blinks whenever the signals transfer from “RF-RECEIVER” to “HT12D”. “HT12D” decodes the information and sends the data to “89s52- MICROCONTROLLER”. 13th, 12th, 11th and 10th out-put pins of “HT12D” are given to “89s52-MICROCONTROLLER” ports as input. The entire operation is done in this “89s52 MICROCONTROLLER”. It takes all the inputs and generates the output.**

**“89s52-MICROCONTROLLER” it is a 40pin IC. 9th pin is connected to +Ve Vcc of “HC05-BLUETOOTH-MODULE” through a 10uf capacitor and reset button. The RXD & TXD are connected to TX & RX pins of “HC-05”. The XTAL1&XTAL2 pins are connected to ground trough two 33pf capacitors. The crystal oscillator which 11.0592MHz is placed between 18th &19th pins. 20th is grounded. The output data from “89s52-MICROCONTROLLER” is connected to “L293D” to four pins. The out-put pins 21 to 24 of “89s52” connected to 15th, 10th, 2nd&7th pins of “L293D” as input. The 14th, 15th pins of “89s52-MICROCONTROLLER” connected to echo and trigger pins of “HLSR04-ULTRASONIC SENSOR”. The ground pin of “HLSR04” is connected to ground pin of “RF-RECEIVER”**

**“L293D” it is just motor driver IC which consists of 14 pins, 4th,5th, 12th&13th pins are ground terminals which are connected to 9th pin of “89s52” through 10K resister. 1st, 9th, 16th of “L293D” are connected to 18th pin of “HT12D” through 31&40 pins of “89s52” in which 10K resister is placed between them. 3rd, 6th, 11th &14th pins are output pins of which 3rd &6th are connected to motor one (M1) and 11th &14th are connected to motortwo (M2). “L293D” follows the instructions given by the “89s52-MICROCONTROLLER” and runs as per the command. This means the motors run to left, right and to & fro directions. The Vs pin of “L293D” is connected to 8volts Vcc. except this Vs pin of “L293D” the rest of all pins require only 5volts.**

**NOTE: The peripherals and pins of “89s52-MICROCONTROLLER” are explained further chapters.**

**A Gesture Controlled robot is a kind of robot which can be controlled by your hand gestures not by old buttons. You just need to wear a small transmitting device in your hand which included an acceleration meter. This will transmit an appropriate command to the robot so that it can do whatever we want. The transmitting device included a comparator IC for analog to digital conversion and an encoder IC (HT12E) which is use to encode the four bit data and then it will transmit by an RF Transmitter module.**

**At the receiving end an RF Receiver module receiver’s the encoded data and decodes it by a decoder IC (HT12D). This data is then processed by a microcontroller (AT89s52) and finally our motor driver to control the motors.**

**We use ultrasonic sensor to measure the distance of an object from the required station. The main purpose of the sensor is to generate high frequency waves and get the information in the form of feedback waves. This feedback waves gives the information about object at what distance it is.**

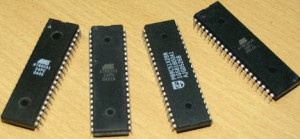
**In this we use a Bluetooth module; it encodes the signals/information from the ultrasonic to the control station. Here we use a special device, Camera which is an optional one. It is specially used for live streaming, sometimes for recording also.**

* **Introduction To Microcontroller :`**

**A microcontroller is a small computer (SoC) on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of Ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.**

**Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems.**

**Some microcontrollers may use four-bit words and operate at clock rate frequencies as low as 4 kHz, for low power consumption (single-digit milliwatts or microwatts). They will generally have the ability to retain functionality while waiting for an event such as a button press or other interrupt; power consumption while sleeping (CPU clock and most peripherals off) may be just nanowatts, making many of them well suited for long lasting battery applications. Other microcontrollers may serve performance-critical roles, where they may need to act more like a digital signal processor (DSP), with higher clock speeds and power consumption.**

****

**Different Types of Microcontrollers**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Microcontroller:** | **Family:** | **No Of Bits:** | **Architecture:** | **Application:** |
| **AT89S52** | **ATMEL** | **8** | **Harvard** | **Low level data handling** |
| **ARM11** | **ARM** | **32** | **Harvard** | **High level data handling** |
| **ATM eg a8** | **AVR** | **8** | **Harvard** | **Low level data handling** |
| **MCS251** | **INTEL8051** | **8** | **VonNeumann** | **Low level data handling** |
| **LPC2138** | **ARM** | **32** | **Harvard** | **High level data handling** |
| **PIC16F877A** | **PIC** | **8** | **Harvard** | **Low level data handling** |

* **Pin description of AT89s52:**

**“8051” is the name of a big family of microcontrollers. The device which we used in our project was the “AT89S52” which is a typical “8051” microcontroller manufactured by Atmel. The block diagram provided by Atmel in their datasheet that showed the architecture of 89S52 device seemed a bit complicated. A simpler architecture can be represented below**

**The “89S52” has 4 different ports, each one having 8 Input/output lines providing a total of 32 I/O lines. Those ports can be used to output DATA and orders do other devices, or to read the state of a sensor, or a switch. Most of the ports of the “89S52” have 'dual function' meaning that they can be used for two different functions.**

**The first one is to perform input/output operations and the second one is used to implement special features of the microcontroller like counting external pulses, interrupting the execution of the program according to external events, performing serial data transfer or connecting the chip to a computer to update the software. Each port has 8 pins, and will be treated from the software point of view as an 8-bit variable called 'register', each bit being connected to a different Input/output pin.**

**There are two different memory types: “RAM” and “EEPROM”. Shortly, “RAM” is used to store variable during program execution, while the “EEPROM” memory is used to store the program itself, that's why it is often referred to as the 'program memory'. It is clear that the CPU (Central Processing Unit) is the heart of the micro controllers. It is the CPU that will Read the program from the FLASH memory and execute it by interacting with the different peripherals.**

* **Block Diagram Of AT89s52:**
* **Pin diagram:**
* **VCC : Supply voltage.**
* **GND: Ground.**
* **Port 0:**

**Port 0 is an 8-bit open drain bidirectional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs. Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pull-ups. Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pull-ups are required during program verification.**

* **Port 1:**

**Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL) because of the internal pull-ups. Port 1 also receives the low-order address bytes during Flash programming and verification.**

****

* **Port 2 :**

**Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (IIL) because of the internal pull-ups. Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.**

* **Port 3:**

**Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (IIL) because of the pull-ups. Port 3 receives some control signals for Flash programming and verification. Port 3 also serves the functions of various special features of the AT89S51, as shown in the following table.**

****

* **RST:**

**Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. This pin drives High for 98 oscillator periods after the Watchdog times out. The DISRTO bit in SFR AUXR (address 8EH) can be used to disable this feature. In the default state of bit DISRTO, the RESET HIGH out feature is enabled.**

* **ALE/PROG:**

**Address Latch Enable (ALE) is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during Flash programming. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory. If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.**

* **PSEN:**

**Program Store Enable (PSEN) is the read strobe to external program memory. When the AT89S51 is executing code from external program memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory.**

* **EA/VPP:**

**External Access Enable. EA must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, EA will be internally latched on reset. EA should be strapped to VCC for internal program executions. This pin also receives the 12-volt programming enable voltage (VPP) during Flash programming.**

* **XTAL1:**

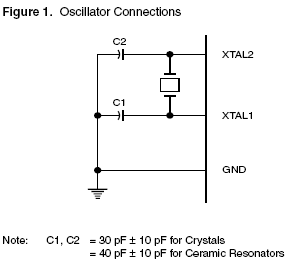
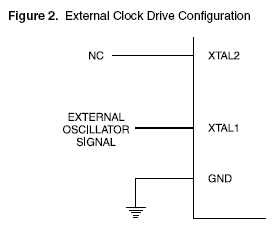
**Input to the inverting oscillator amplifier and input to the internal clock operating circuit.**

* **XTAL2:**

**Output from the inverting oscillator amplifier.**

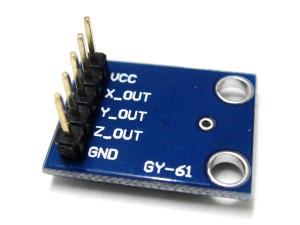
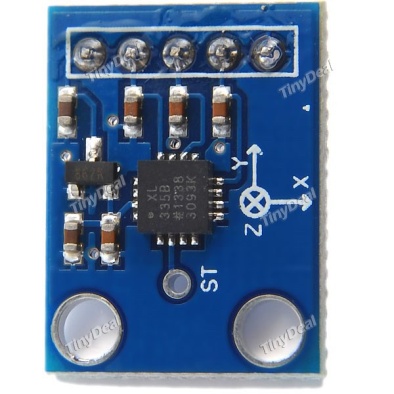
* **Oscillator Characteristics:**

**XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figs 6.2.3. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 6.2.4.There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.**

**Fig 1. Oscillator Connections Fig 2. External Clock Drive Configuration**

* **Features:**
* **Compatible with MCS-51 ® Products**
* **8K Bytes of In-System Programmable (ISP) Flash Memory**
* **Endurance: 1000 Write/Erase Cycles**
* **4.0V to 5.5V Operating Range**
* **Fully Static Operation: 0 Hz to 33 MHz**
* **Three-level Program Memory Lock**
* **256 x 8-bit Internal RAM**
* **32 Programmable I/O Lines**
* **Three 16-bit Timer/Counters**
* **Eight Interrupt Sources**
* **Full Duplex UART Serial Channel**
* **Low-power Idle and Power-down Modes**
* **Interrupt Recovery from Power-down Mode**
* **Watchdog Timer**
* **Dual Data Pointer**
* **Power-off Flag**
* **ADXL335:**

****

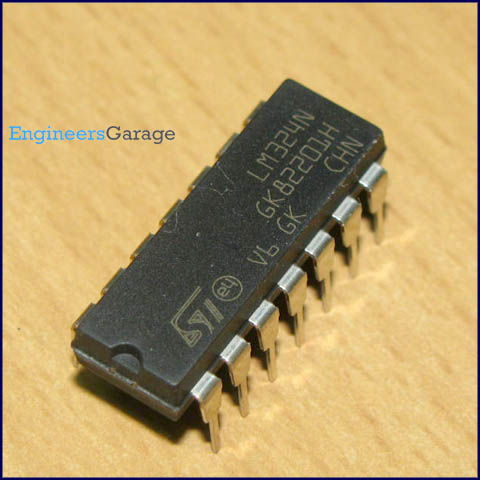
* **Description:**

**The ADXL335 is the latest and greatest from simulation equipment， known for their exceptional quality MEMS devices.**

**It is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs. The product measures acceleration with a minimum full-scale range of ±3g. It can measure the static acceleration of gravity in tilt-sensing applications, and dynamic acceleration resulting from motion, shock, or vibration.**

**The user selects the bandwidth of the accelerometer using the CX, CY, and CZ capacitors at the XOUT, YOUT, and ZOUT pins. Bandwidths can be to suit the application.**

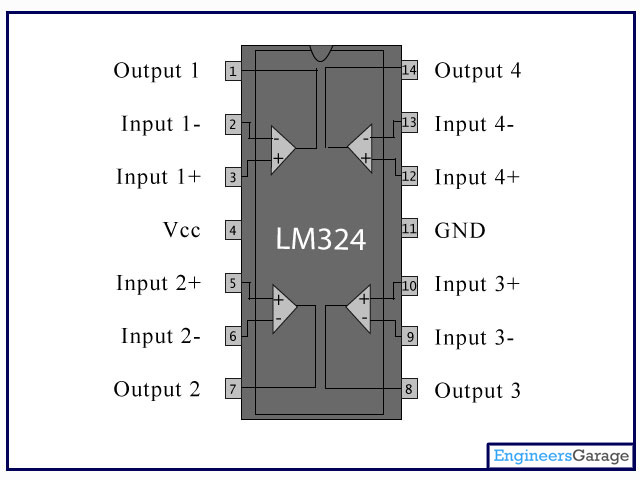
* **Specifications:**
* **Wide power range DC3V to 5V**
* **Small, low-profile package**
* **Low power 350μA at 3V (typical)**
* **High sensitive**
* **10,000 g shock survival**
* **BW adjustment with a single capacitor per axis**
* **LM324:**

****

* **Datasheet:**

**LM324 is a 14pin IC consisting of four independent operational amplifiers (op-amps) compensated in a single package. Op-amps are high gain electronic voltage amplifier with differential input and, usually, a single-ended output. The output voltage is many times higher than the voltage difference between input terminals of an op-amp.**

**These op-amps are operated by a single power supply LM324 and need for a dual supply is eliminated. They can be used as amplifiers, comparators, oscillators, rectifiers etc. The conventional op-amp applications can be more easily implemented with LM324.**

****

* **Pin Description:**

|  |  |  |
| --- | --- | --- |
| PIN. No: | FUNCATION: | NAMES: |
| 1 | **Output of 1st comparator** | **Output 1** |
| 2 | **Inverting input of 1st comparator** | **Input 1-** |
| 3 | **Non-inverting input of 1st comparator** | **Input 1+** |
| 4 | **Supply voltage; 5V (up to 32V)** | **Vcc** |
| 5 | **Non-inverting input of 2nd comparator** | **Input 2+** |
| 6 | **Inverting input of 2nd comparator** | **Input 2-** |
| 7 | **Output of 2nd comparator** | **Output 2** |
| 8 | **Output of 3rd comparator** | **Output 3** |
| 9 | **Inverting input of 3rd comparator** | **Input 3-** |
| 10 | **Non-inverting input of 3rd comparator** | **Input 3+** |
| 11 | **Ground (0V)** | **Ground** |
| 12 | **Non-inverting input of 4th comparator** | **Input 4+** |
| 13 | **Inverting input of 4th comparator** | **Input 4-** |
| 14 | **Output of 4th comparator** | **Output 4** |

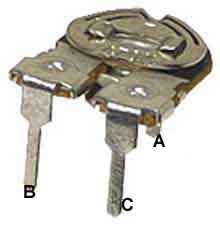
* **10K VARIABLE RESISTOR:**

****

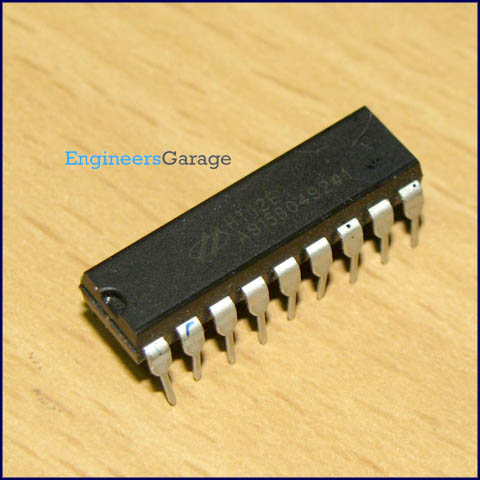
## How to use a preset?

**There are 3 pins/terminals on a preset. The maximum resistance that a preset can provide is written on it. If 100K is written on preset, it means that we can vary its resistance from 0 Ohm to 100K. A movable metal is rotated in clockwise or anticlockwise direction that changes the resistance of preset.**

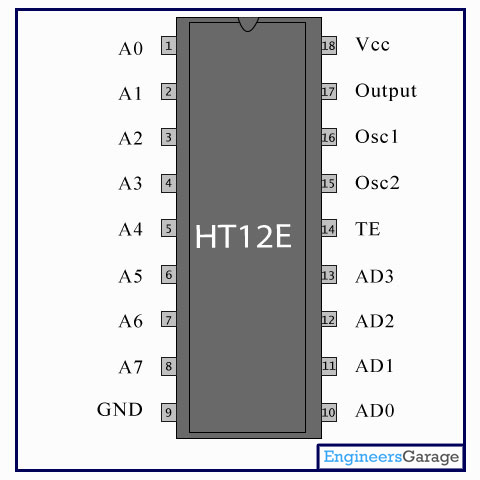
**Now, we name the three terminals as A, B, C.**

**[](http://www.buildcircuit.com/wp-content/uploads/2010/11/preset1.jpg)[http://www.buildcircuit.com/wp-content/uploads/2010/11/preset.gif](http://www.buildcircuit.com/wp-content/uploads/2010/11/preset.gif)**

* **If we take terminal A and terminal B, and rotate the movable metal in clockwise direction, the resistance of preset increases from 0 to maximum. As we move the metal in anticlockwise direction, the resistance decreases.**
* **If we take terminal A and terminal C, and rotate the movable metal in anticlockwise direction, the resistance of preset increases from 0 to maximum. As we move the metal in clockwise direction, the resistance decreases.**
* **HT12E:**

****

* **Datasheet:**

****

**HT12E is an encoder integrated circuit of 212 series of encoders. They are paired with 212 series of decoders for use in remote control system applications. It is mainly used in interfacing RF and infrared circuits. The chosen pair of encoder/decoder should have same number of addresses and data format.**

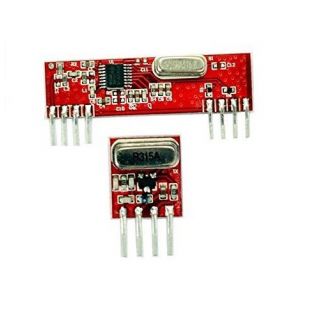
**Simply put, HT12E converts the parallel inputs into serial output. It encodes the 12 bit parallel data into serial for transmission through an RF transmitter. These 12 bits are divided into 8 address bits and 4 data bits.**

**HT12E has a transmission enable pin which is active low. When a trigger signal is received on TE pin, the programmed addresses/data are transmitted together with the header bits via an RF or an infrared transmission medium. HT12E begins a 4-word transmission cycle upon receipt of a transmission enable. This cycle is repeated as long as TE is kept low. As soon as TE returns to high, the encoder output completes its final cycle and then stops.**

* **Pin Description:**

|  |  |  |
| --- | --- | --- |
| **PIN. No:** | **FUNCATION:** | **NAMES:** |
| **1** | **8 bit Address pins for input** | **A0** |
| **2** | **A1** |
| **3** | **A2** |
| **4** | **A3** |
| **5** | **A4** |
| **6** | **A5** |
| **7** | **A6** |
| **8** | **A7** |
| **9** | **Ground (0V)** | **Ground** |
| **10** | **4 bit Data/Address pins for input** | **AD0** |
| **11** | **AD1** |
| **12** | **AD2** |
| **13** | **AD3** |
| **14** | **Transmission enable; active low** | **TE** |
| **15** | **Oscillator input** | **Osc2** |
| **16** | **Oscillator output** | **Osc1** |
| **17** | **Serial data output** | **Output** |
| **18** | **Supply voltage; 5V (2.4V-12V)** | **Vcc** |

* **RF TRANSMITTER AND RECEIVER:**

****

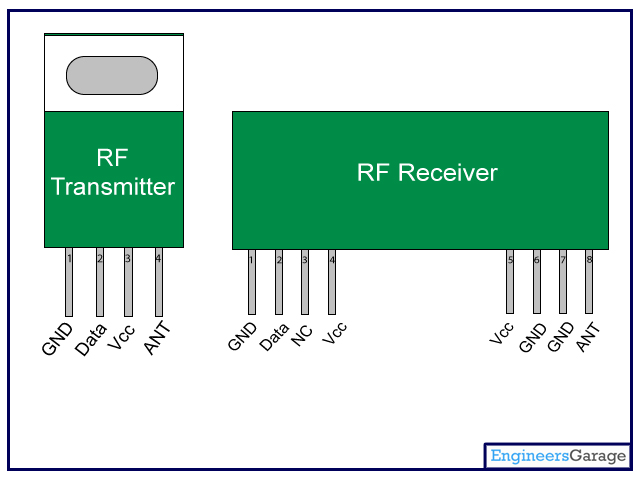
* **Datasheet:**

**The RF module, as the name suggests, operates at Radio Frequency. The corresponding frequency range varies between 30 kHz & 300 GHz. In this RF system, the digital data is represented as variations in the amplitude of carrier wave. This kind of modulation is known as Amplitude Shift Keying (ASK).**

**Transmission through RF is better than IR (infrared) because of many reasons. Firstly, signals through RF can travel through larger distances making it suitable for long range applications. Also, while IR mostly operates in line-of-sight mode, RF signals can travel even when there is an obstruction between transmitter & receiver. Next, RF transmission is more strong and reliable than IR transmission. RF communication uses a specific frequency unlike IR signals which are affected by other IR emitting sources.**

**This RF module comprises of an RF Transmitter and an RF Receiver. The transmitter/receiver (Tx/Rx) pair operates at a frequency of 434 MHz. An RF transmitter receives serial data and transmits it wirelessly through RF through its antenna connected at pin4. The transmission occurs at the rate of 1Kbps - 10Kbps.The transmitted data is received by an RF receiver operating at the same frequency as that of the transmitter.**

**The RF module is often used along with a pair of encoder/decoder. The encoder is used for encoding parallel data for transmission feed while reception is decoded by a decoder. HT12E-HT12D, HT640-HT648, etc. are some commonly used encoder/decoder pair ICs.**

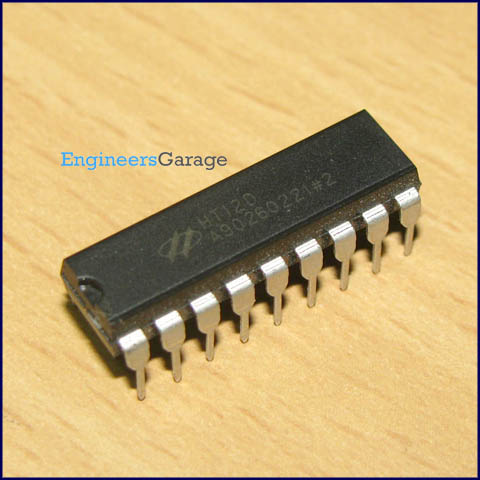
****

* **Pin Description:**

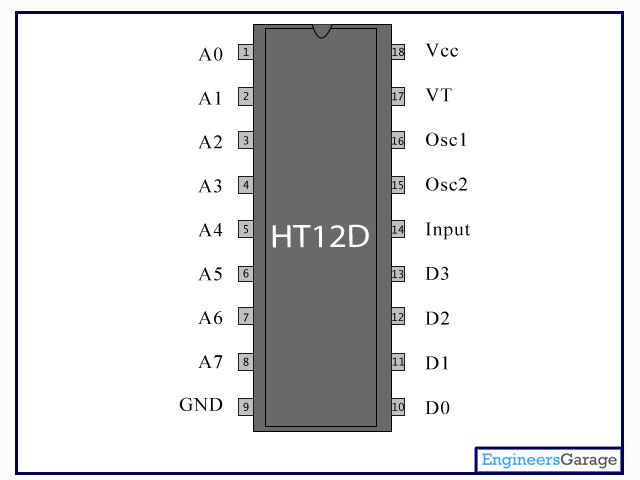
|  |  |  |
| --- | --- | --- |
| PIN No: | FUNCTION: | NAME: |
| 1 | **Ground (0V)** | **Ground** |
| 2 | **Serial data input pin** | **Data** |
| 3 | **Supply voltage; 5V** | **Vcc** |
| 4 | **Antenna output pin** | **ANT** |

|  |  |  |
| --- | --- | --- |
| PIN No: | FUNCTION: | NAME: |
| 1 | **Ground (0V)** | **Ground** |
| 2 | **Serial data input pin** | **Data** |
| 3 | **Linear output pin; not connected** | **NC** |
| 4 | **Supply voltage; 5V** | **VCC** |
| 5 | **Supply voltage; 5V** | **VCC** |
| 6 | **Ground (0V)** | **GROUND** |
| 7 | **Ground (0V)** | **GROUND** |
| 8 | **Antenna input pin** | **ANT** |

* **HT12D:**

****

* **Datasheet**

****

**HT12D is a decoder integrated circuit that belongs to 212 series of decoders. This series of decoders are mainly used for remote control system applications, like burglar alarm, car door controller, security system etc. It is mainly provided to interface RF and infrared circuits.  They are paired with 212 series of encoders. The chosen pair of encoder/decoder should have same number of addresses and data format.**

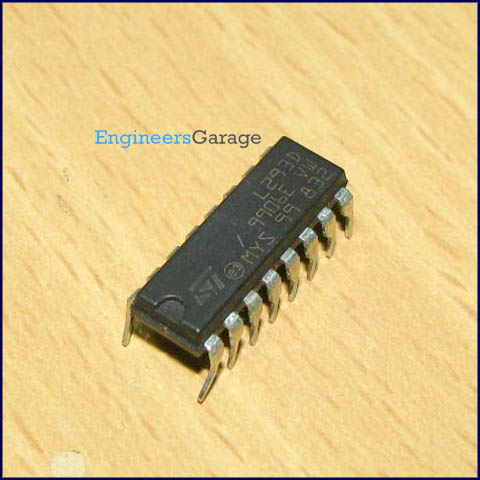
**In simple terms, HT12D converts the serial input into parallel outputs. It decodes the serial addresses and data received by, say, an RF receiver, into parallel data and sends them to output data pins. The serial input data is compared with the local addresses three times continuously. The input data code is decoded when no error or unmatched codes are found. A valid transmission in indicated by a high signal at VT pin.**

**HT12D is capable of decoding 12 bits, of which 8 are address bits and 4 are data bits. The data on 4 bit latch type output pins remain unchanged until new is received.**

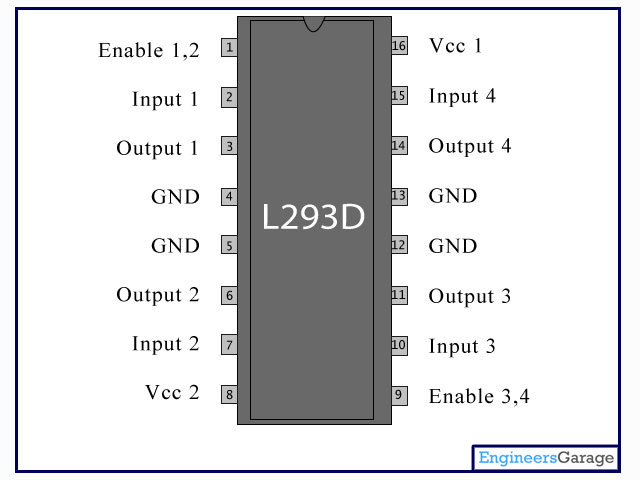
* **Pin Description:**

|  |  |  |
| --- | --- | --- |
| **PIN. No:** | **FUNCATION:** | **NAMES:** |
| **1** | **8 bit Address pins for input** | **A0** |
| **2** | **A1** |
| **3** | **A2** |
| **4** | **A3** |
| **5** | **A4** |
| **6** | **A5** |
| **7** | **A6** |
| **8** | **A7** |
| **9** | **Ground (0V)** | **Ground** |
| **10** | **4 bit Data/Address pins for input** | **D0** |
| **11** | **D1** |
| **12** | **D2** |
| **13** | **D3** |
| **14** | **Serial data input** | **INPUT** |
| **15** | **Oscillator output** | **Osc2** |
| **16** | **Oscillator input** | **Osc1** |
| **17** | **Valid transmission; active high** | **VT** |
| **18** | **Supply voltage; 5V (2.4V-12V)** | **Vcc** |

* **LT93D:**

****

* **Datasheet:**

****

**L293D is a dual H-bridge motor driver integrated circuit (IC). Motor drivers act as current amplifiers since they take a low-current control signal and provide a higher-current signal. This higher current signal is used to drive the motors.**

**L293D contains two inbuilt H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The motor operations of two motors can be controlled by input logic at pins 2 & 7 and 10 & 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions, respectively.**

**Enable pins 1 and 9 (corresponding to the two motors) must be high for motors to start operating. When an enable input is high, the associated driver gets enabled. As a result, the outputs become active and work in phase with their inputs. Similarly, when the enable input is low, that driver is disabled, and their outputs are off and in the high-impedance state.**

* **Pin Description:**

|  |  |  |
| --- | --- | --- |
| **PIN. No:** | **FUNCATION:** | **NAMES:** |
| **1** | **Enable pin for Motor 1; active high** | **Enable 1,2** |
| **2** | **Input 1 for Motor 1** | **Input 1** |
| **3** | **Output 1 for Motor 1** | **Output 1** |
| **4** | **Ground (0V)** | **Ground** |
| **5** | **Ground (0V)** | **Ground** |
| **6** | **Output 2 for Motor 1** | **Output 2** |
| **7** | **Input 2 for Motor 1** | **Input 2** |
| **8** | **Supply voltage for Motors; 9-12V (up to 36V)** | **Vcc 2** |
| **9** | **Enable pin for Motor 2; active high** | **Enable 3,4** |
| **10** | **Input 1 for Motor 1** | **Input 3** |
| **11** | **Output 1 for Motor 1** | **Output 3** |
| **12** | **Ground (0V)** | **Ground** |
| **13** | **Ground (0V)** | **Ground** |
| **14** | **Output 2 for Motor 1** | **Output 4** |
| **15** | **Input2 for Motor 1** | **Input 4** |
| **16** | **Supply voltage; 5V (up to 36V)** | **Vcc 1** |

* **Gear Motors:**

****

* **Datasheet:**

**300 RPM Side Shaft Heavy Duty DC Gear Motor is suitable for large robots / automation systems. It has sturdy construction with gear box built to handle stall torque produced by the motor. Drive shaft is supported from both sides with metal bushes. Motor runs smoothly from 4V to 12V and gives 300 RPM at 12V. Motor has 8mm diameter, 17.5mm length drive shaft with D shape for excellent coupling.**

**Table below gives fairly good idea of the motor’s performance in terms of RPM vs voltage at no load and that of stall torque at different voltages.**

**Note: This motor will be bit noisy while running.**

* **Specifications:**
* **RPM: 300 at 12V**
* **Voltage: 4V to 12V**
* **Stall torque: 23Kg-cm at stall current of 8.4A@12V**
* **Shaft diameter: 8mm**
* **Shaft length: 17.5mm**
* **Gear assembly: Spur**
* **Brush type: Carbon**
* **Motor weight: 280gms**
* **Dimension: Refer to diagram below**
* **HC-SR04 Ultra Sonic Sensor:**

****

* **Datasheet:**

**HC-SR04 based Distance Finder gives the distance from an obstacle in centimeters. It has a range of 2cms to 400cms. The project is build around AT89S52 which 8051 based microcontroller. The distance is displayed on Seven Segment Displays. The working of the ultrasonic sensors is quite simple and they are easy to interface with the microcontroller.**

**The sensor module has 4-pins out of which Pin-1 and Pin-4 are +Vcc and Gnd respectively.Pin-2 is Trigger and Pin-3 is Echo pin. When a High pulse of 10us is applied at TRIG pin, the ultrasonic transmitter sends 8 consecutive pulses of 40 kHz frequency. As the Eighth pulse is sent the ECHO pin of the sensor becomes HIGH. Now when the ultrasonic waves reflect from any surface and are received by the Receiver, the ECHO pin becomes LOW.**

* **HL-05 Bluetooth Module:**

****

* **Datasheet:**

**The Bluetooth Module Breakout is the latest Bluetooth wireless serial cable! This version of the popular Bluetooth uses the HC-05/HC-06 module. These modems work as a serial (RX/TX) pipe. Any serial stream from 9600 to 115200bps can be passed seamlessly from your computer to your target.**

**The remote unit can be powered from 3.3V up to 6V for easy battery attachment. All signal pins on the remote unit are 3V-6V tolerant. No level shifting is required. Do not attach this device directly to a serial port. You will need an RS232 to TTL converter circuit or Arduio XBee USB Adapter if you need to attach this to a computer. You can either solder a 6-pin header or individual wires.**

**Unit comes without a connector. Please see related male and female pins below.**

**And now, we provide HC-05/06, HC-05 could be setting to Master or Slave by user. HC-06 just is Master or Slave that could be customized. This module default is HC-06 with Slave mode.**

**Note: Be paired must Master devices and Slave devices each other. Master and Master devices, Slave and Slave devices can't pair each other. The HC-06 (just supply slave mode and more easy) Slave mode could paired with PDA, PC etc.**

* **Specifications:**
* **Bluetooth protocol:  Bluetooth Specification v2.0+EDR**
* **Frequency:  2.4GHz ISM band**
* **Modulation:  GFSK(Gaussian Frequency Shift Keying)**
* **Emission power:  ≤4dBm, Class 2**
* **Sensitivity:  ≤-84dBm at 0.1% BER**
* **Speed: Asynchronous:  2.1Mbps(Max) / 160 kbps, Synchronous: 1Mbps/1Mbps**
* **Security:  Authentication and encryption**
* **Profiles:  Bluetooth serial port**
* **Power supply: +3.3VDC 50mA**
* **Working temperature: -20 ~ +75 Centigrade**
* **Dimensions: 15.2x35.7x5.6mm**
* **WIRELESS CAMERE WITH MODULE:**

****

* **Datasheet:**

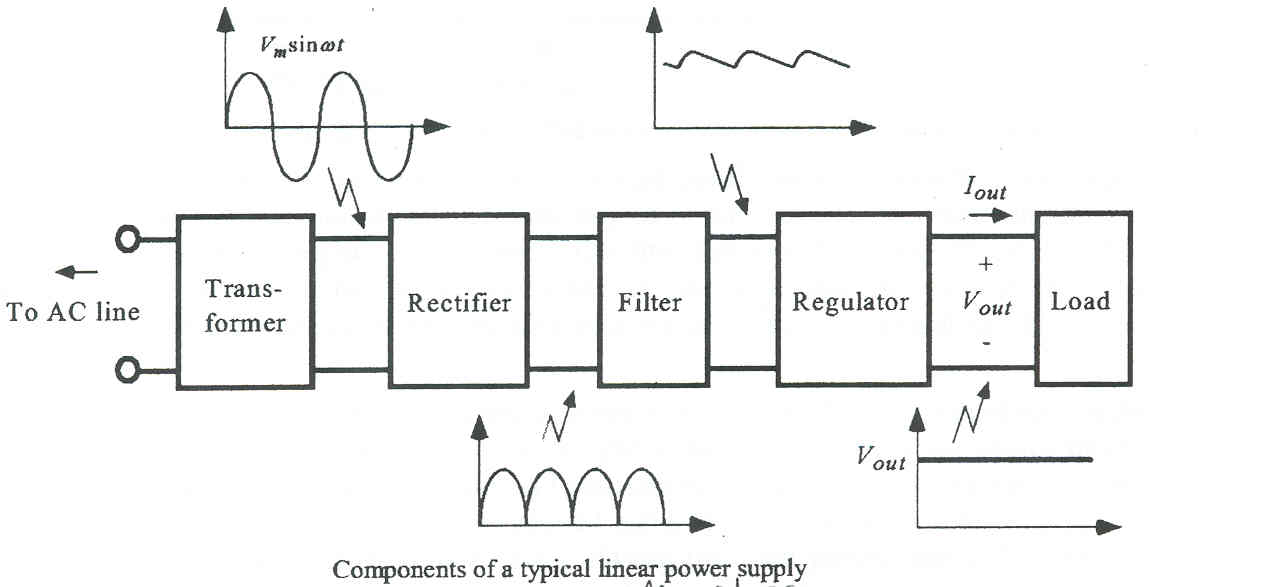
**This is a low cost wireless camera suitable for 50 to 100 meters wireless communication range. It is most suitable for small sized mobile robots. You can transmit wireless image acquired by the robot to the PC and do the image processing as well as robot control. It can work in wireless and wired mode. Wireless camera can be powered by 9V battery or 9V DC adapter. Receiver is powered by 9V / 12V DC adapter.**

**Important: Color image acquire by the camera is not very high quality and depending on the display device used it may show faint horizontal lines in the video feed.**

* **Specifications:**
* **Transmission Signal: Video, Audio**
* **Linear Transmission Distance: 50-100m**
* **Voltage: DC+9V**
* **Current: 300Ma**
* **Output Electrical Level: 50mW**
* **Output Frequency: 1.2G/2.4G**
* **Receiving Frequency: 1.2G/2.4G**
* **Antenna:50ohm SMA**
* **Power supply :**

**The power supplies are designed to convert high voltage AC mains electricity to a suitable low voltage supply for electronic circuits and other devices. A power supply can by broken down into a series of blocks, each of which performs a particular function. A D.C. power supply which maintains the output voltage constant irrespective of A.C. mains fluctuations or load variations is known as “Regulated D.C Power Supply”**

* **For example a 5V regulated power supply system as shown below:**



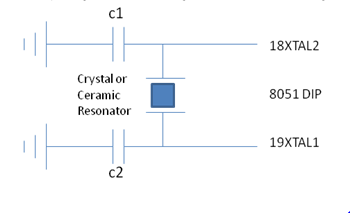
* **Crystal oscillator :**

**A crystal oscillator is an electronic oscillator circuit that uses the mechanical resonance of a vibrating crystal of piezoelectric material to create an electrical signal with a very precise frequency. This frequency is commonly used to keep track of time (as in quartz wristwatches), to provide a stable clock signal for digital integrated circuits, and to stabilize frequencies for radio transmitters and receivers. The most common type of piezoelectric resonator used is the quartz crystal, so oscillator circuits designed around them became known as "crystal oscillators."**

**In this project 12MHz frequency circuit is to be used and placed in between pin 18 – pin 19 i.e., XTAL1 and XTAL2 which generates oscillating frequency. It is the heart of the microcontroller that generates the clock pulses to synchronize all internal operations. Microcontrollers can run at specified max and min frequencies, typically 1MHz to 16MHz. Generally some internal memories are dynamic, it can be operate above Min frequencies otherwise data will be lost. The smallest interval of time to accomplish any simple instruction or a part of complex instruction known as “Machine Cycle”. The time of any particular instruction is given by**

**Tinst = C x 12d/crystal freq**

**11.0592 MHz is the exact freq for standard baud rates (300, 1200, 2400, 4800, and 9600).**



* **Introduction to Micro vision Keil (IDE):-**

**Keil was founded in 1982 by Günter and Reinhard Keil, initially as a German GbR. In April 1985 the company was converted to *Keil Elektronik GmbH* to market add-on products for the development tools provided by many of the silicon vendors. Keil implemented the first C compiler designed from the ground-up specifically for the 8051 microcontroller.**

**Keil is a cross compiler. So first we have to understand the concept of compilers and cross compilers. After then we shall learn how to work with keil.**

* **Concept of compiler:-**

**Compilers are programs used to convert a High Level Language to object code. Desktop compilers produce an output object code for the underlying microprocessor, but not for other microprocessors. I.E the programs written in one of the HLL like ‘C’ will compile the code to run on the system for a particular processor like x86 (underlying microprocessor in the computer). For example compilers for Dos platform is different from the Compilers for UNIX platform**

**So if one wants to define a compiler then compiler is a program that translates source code into object code. The compiler derives its name from the way it works, looking at the entire piece of source code and collecting and reorganizing the instruction. See there is a bit little difference between compiler and an interpreter. Interpreter just interprets whole program at a time while compiler analyzes and execute each line of source code in succession, without looking at the entire program.**

**The advantage of interpreters is that they can execute a program immediately. Secondly programs produced by compilers run much faster than the same programs executed by an interpreter. However compilers require some time before an executable program emerges. Now as compilers translate source code into object code, which is unique for each type of computer, many compilers are available for the same language.**

* **Concept of cross compiler:-**

**A cross compiler is similar to the compilers but we write a program for the target processor (like 8051 and its derivatives) on the host processors (like computer of x86)**

**It means being in one environment you are writing a code for another environment is called cross development. And the compiler used for cross development is called cross compiler**

**So the definition of cross compiler is a compiler that runs on one computer but produces object code for a different type of computer. Cross compilers are used to generate software that can run on computers with a new architecture or on special-purpose devices that cannot host their own compilers. Cross compilers are very popular for embedded development, where the target probably couldn't run a compiler. Typically an embedded platform has restricted RAM, no hard disk, and limited I/O capability. Code can be edited and compiled on a fast host machine (such as a PC or UNIX workstation) and the resulting executable code can then be downloaded to the target to be tested. Cross compilers are beneficial whenever the host machine has more resources (memory, disk, I/O etc) than the target.  Keil C Compiler is one such compiler that supports a huge number of host and target combinations. It supports as a target to 8 bit microcontrollers like Atmel and Motorola etc.**

* **Why do we need cross compiler?**

**There are several advantages of using cross compiler. Some of them are described as follows**

* **By using this compilers not only can development of complex embedded systems be completed in a fraction of the time, but reliability is improved, and maintenance is easy.**
* **Knowledge of the processor instruction set is not required.**
* **A rudimentary knowledge of the 8051’s memory architecture is desirable but not necessary.**
* **Register allocation and addressing mode details are managed by the compiler.**
* **The ability to combine variable selection with specific operations improves program readability.**
* **Keywords and operational functions that more nearly resemble the human thought process can be used.**
* **Program development and debugging times are dramatically reduced when compared to assembly language programming.**
* **The library files that are supplied provide many standard routines (such as formatted output, data conversions, and floating-point arithmetic) that may be incorporated into your application.**
* **Existing routine can be reused in new programs by utilizing the modular programming techniques available with C.**
* **The C language is very portable and very popular. C compilers are available for almost all target systems. Existing software investments can be quickly and easily converted from or adapted to other processors or environments.**

**Now after going through the concept of compiler and cross compilers lets we start with Keil C cross compiler.**

* **Keil C cross compiler:**

**Keil is a German based Software development company. It provides several development tools like**

**•         IDE (Integrated Development environment)**

**•         Project Manager**

**•         Simulator**

**•         Debugger**

**•         C Cross Compiler, Cross Assembler, Locator/Linker**

**Keil Software provides you with software development tools for the 8051 family of microcontrollers. With these tools, you can generate embedded applications for the multitude of 8051 derivatives. Keil provides following tools for 8051 development**

**1.     C51 Optimizing C Cross Compiler,**

**2.     A51 Macro Assembler,**

**3.     8051 Utilities (linker, object file converter, library manager),**

**4.     Source-Level Debugger/Simulator,**

**5.     µVision for Windows Integrated Development Environment.**

**The keil 8051 tool kit includes three main tools, assembler, compiler and linker.**

**An assembler is used to assemble your 8051 assembly program**

**A compiler is used to compile your C source code into an object file**

**A linker is used to create an absolute object module suitable for your in-circuit emulator.**

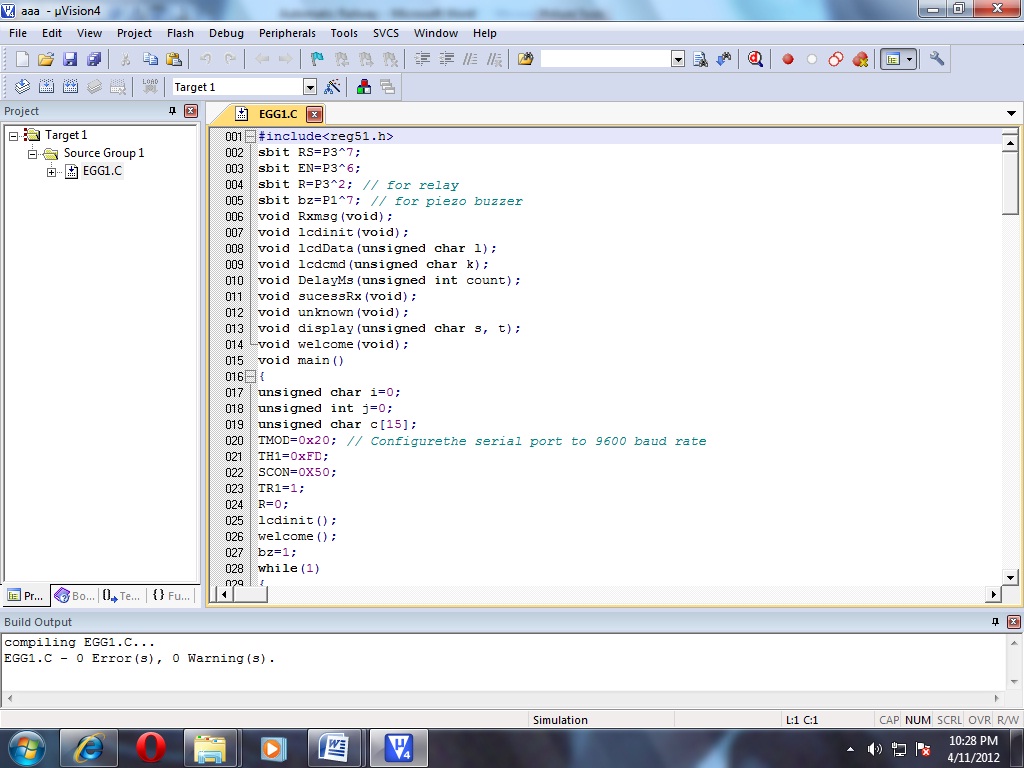
* **8051 project development cycle:**

**These are the steps to develop 8051 project using keil**

1. **Create source files in C or assembly.**
2. **Compile or assemble source files.**
3. **Correct errors in source files.**
4. **Link object files from compiler and assembler.**
5. **Test linked application.**

* **Working with keil :**

**To open keil software click on start menu then program and then select keil2 (or any other version keil3 etc. here the discussion is on keil2 only). Following window will appear on your screen You can see three different windows in this screen.**



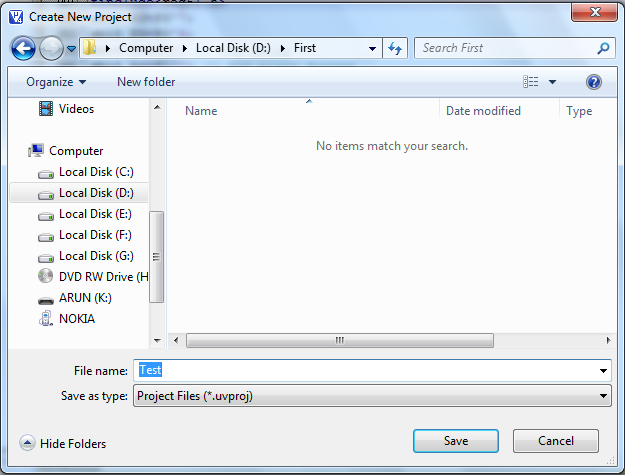
1. **Project work space window**
2. **Editing window**
3. **Output window.**

**Project workspace window is for showing all the related files connected with your project.**

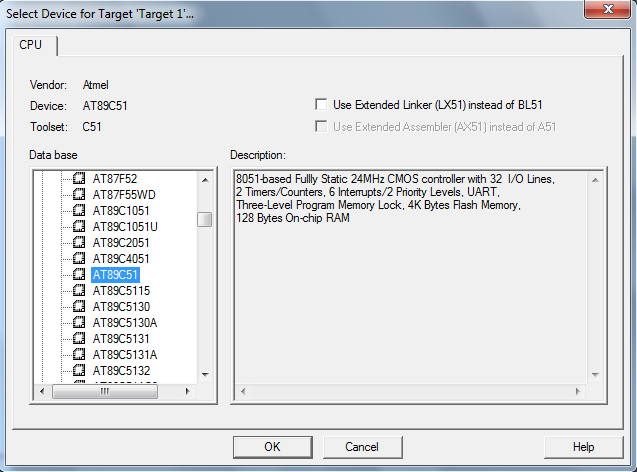
**Editing window is the place where you will edit the code**

**Output window will show the output when you compile or build or run your project.**

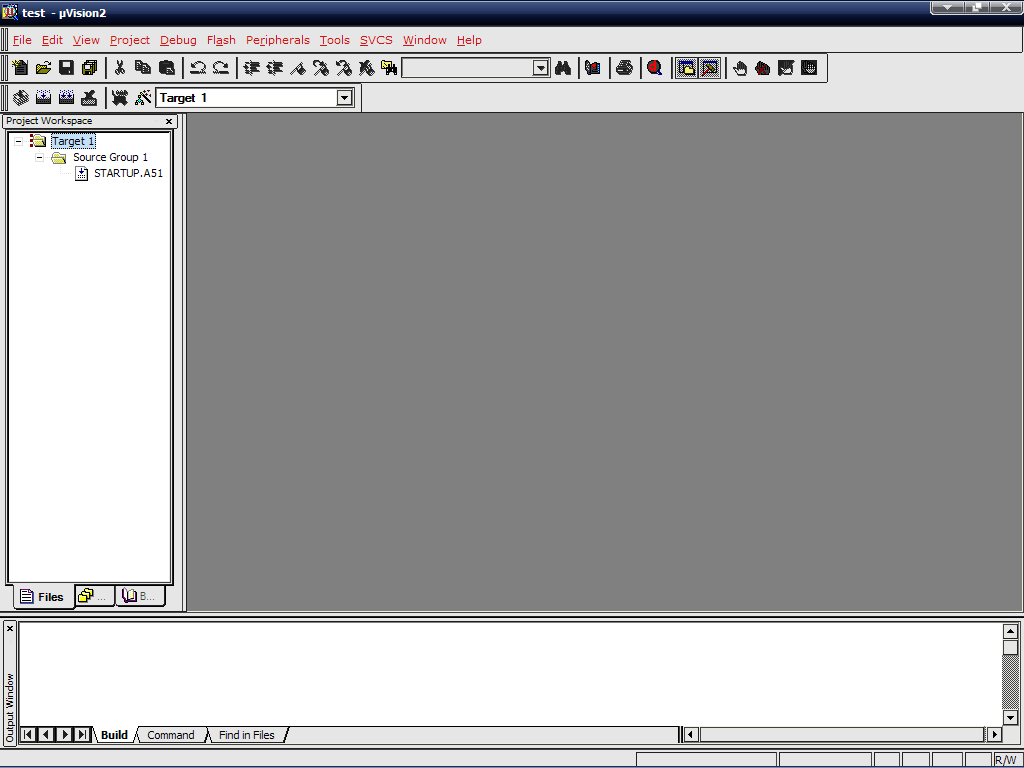
* **Now to start with new project follow the steps**:
* **click on project menu and select new project**
* **you will be asked to create new project in specific directory**
* **Just move to your desired directory and there create a new folder for your project named "first". Here I am creating new project in d:\keil2\myprojects\first as shown in figure**



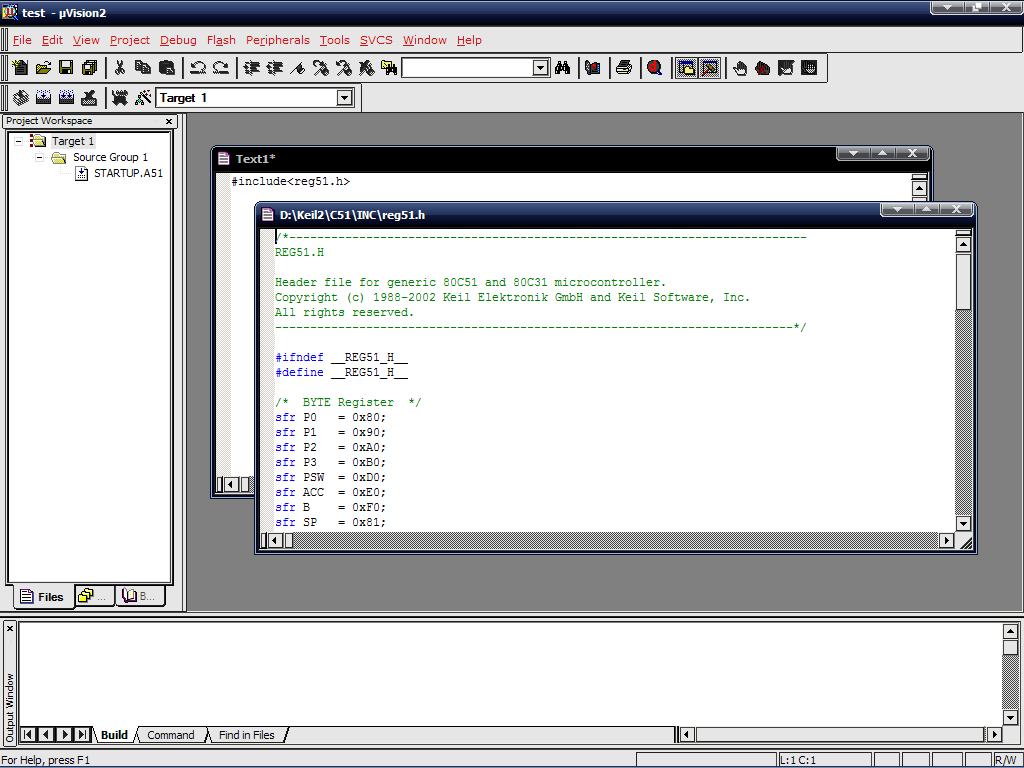
* **Give the name of project as "test". By default it will be saved as \*.v2 extension.**
* **Now you will be asked to choose your target device for which you want to write the program.**
* **Scroll down the cursor and select generic from list. expand the list and select 8051 (all variants)**



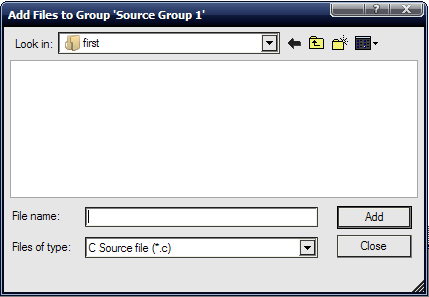
* **When you click OK, you will be asked to add startup code and file to your project folder. Click yes. Now on your screen expand target1 list fully. You will see following window.**



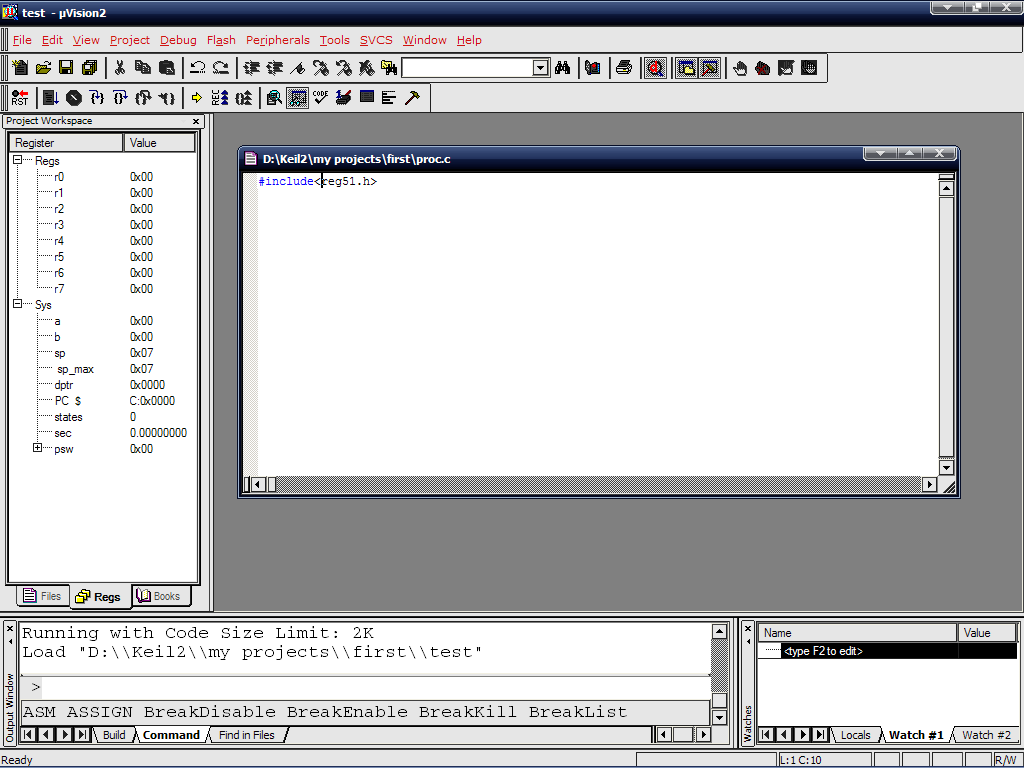
* **Now click on file menu and select new file. editor window will open. Now you can start writing your code.**
* **As you start writing program in C, same way here also you have to first include the header file. Because our target is 8051 our header file will be "reg51.h"**
* **After including this file. just right click on the file and select open document <reg51.h>. The following window will appear**



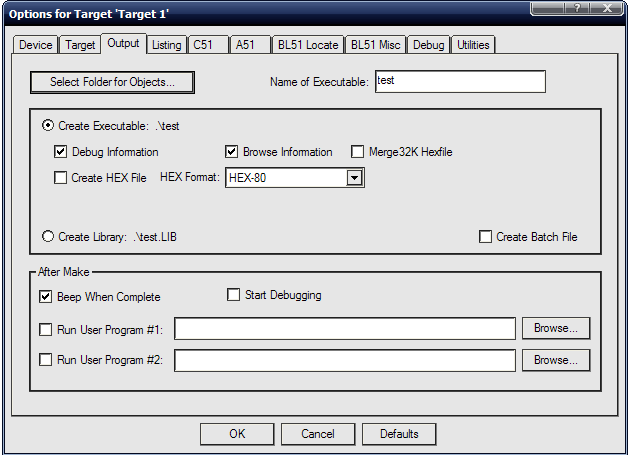
* **If you scroll down cursor you will see that all the SFRs like P0-P3, TCON, and TMOD, ACC, bit registers and byte registers are already defined in this header file. so one can directly use these register names in coding**
* **now you can write your program same as c language starting with void main()**
* **After completing the code save the file in project folder with ".c" extension.**
* **Now right click on "source group 1" in project workspace window. select "add files to source group 1"**
* **select the C file you have created and click add button**



* **you will see that the c file has been added in source group**
* **Now to compile the program from project menu select "build target". In the output window you will see the progress**
* **If there is any compilation error then target will not be created. Remove all the errors and again build the target till you find "0 Error(s)"**
* **Now you are ready to run your program. from debug menu select "start/stop debug session"**
* **You will see your project workspace window now shows most of the SFRs as well as GPRs r0-r7. Also one more window is now opened named "watches". In this window you can see different variable values.**



* **To add variable in watch window go to "watch#1" tab. then type F2 to edit and type the name of your variable**
* **If you want to see the output on ports go to peripheral menu and select I/O ports. Select the desire port. You can give input to port pins by checking or un-checking any check box. Here the check mark means digit 1 and no check mark means 0. the output on the pin will be shown in same manner**
* **To run the program you can use any of the option provided "go", "step by step", "step forward", "step over" etc.**
* **Now after testing your program you need to down load this program on your target board that is 8051. for this you have to create hex file**
* **To create hex file first stop debug session. Again you will be diverted to project workspace window.**
* **Right click on "target 1" and select "option for target 1". Following window will appear.**



* **Select output tag and check "create hex file" box**
* **Now when you again build your program you will see the message in output window "hex file is created".**
* **In your project folder you can see the hex file with same name of your project as "test.hex".**
* **This file you can directly load in 8051 target board and run the application on actual environment.**

**So here the procedure to create a project in keil for 8051 micro controller has been described. To see some sample programs for 8051 in keil just go through the link "sample programs in keil" so that you can get the idea how to write a program for 8051 in keil C.**

* **Source Code:**

**#include<reg51.h>**

**#include"lcd4.h"**

**#include"gsm.h"**

**sbit Trig = P3^5;**

**sbit Echo = P3^4;**

**unsigned char msg1;**

**void init\_serial(void);**

**void DelayMs(unsigned int);**

**void find\_distance();**

**void main()**

**{**

**lcd\_init();**

**init\_serial();**

**Echo = Trig = 0;**

**TMOD |= 0x01;//Timer 0 in 16-bit mode**

**P1=0xf0; // set port as input port**

**P2=0x00; // set port as output port**

**DelayMs(100);**

**while(1)**

**{**

**if(P1==0xf8)**

**{**

**P2=0x0a; //00001010 front**

**}**

**else if(P1==0xf1)**

**{**

**P2=0x02; //00000010**

**}**

**else if(P1==0xf2)**

**{**

**P2=0x08; //00001000**

**}**

**else if(P1==0xf4)**

**{**

**P2=0x05; //00000101 back**

**}**

**else if(msg1=='\*')**

**{**

**find\_distance();**

**msg1=0;**

**}**

**else**

**{**

**P2=0x00;**

**}**

**}**

**}**

**void init\_serial()**

**{**

**TMOD=0x20;**

**TH1=0xFD;**

**SCON=0X50;**

**IE=0X90;**

**TR1=1;**

**}**

**void receive\_data() interrupt 4**

**{**

**while(RI)**

**{**

**msg1=SBUF;**

**RI=0;**

**}**

**}**

**void delay\_us(unsigned int us)//This function provide delay in us uS.**

**{**

**while(us--);**

**}**

**void DelayMs(unsigned int ms)**

**{ // mSec Delay 11.0592 Mhz**

**unsigned int i,j;**

**for(i=0;i<ms;i++)**

**for(j=0;j<127;j++);**

**}**

**void find\_distance()**

**{**

**unsigned int Count,Time,Distance;**

**Trig = 1;//Send 10us start pulse to HC-SR04 as per datasheet of HC-SR04**

**delay\_us(10);//~10us delay**

**Trig = 0;**

**while(Echo == 0);//Wait for Rising edge at Echo pin**

**TR0=1;//Start Timer**

**TL0=TH0=0;//Clear timer count register**

**while(Echo == 1)//Wait for Falling edge at Echo pin**

**{**

**if(TF0 == 1)//timer over if no obstacle is detected**

**break;**

**}**

**TR0=0;//Stop Timer.**

**TF0=0;//clear Timer Over Flow Flag**

**Count = TL0 + TH0\*256;//Calculate number of count**

**Time = Count\*1.085;//Calculate total time in uS.**

**Distance = Time/58;//As per datasheet of HC-SR04 Distance is in Centimeter**

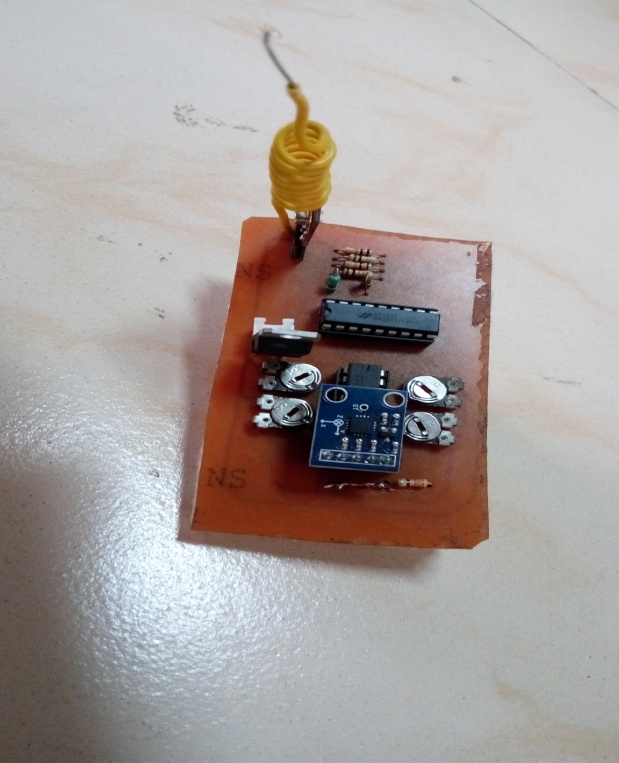
**TX\_String("Distance: ");**

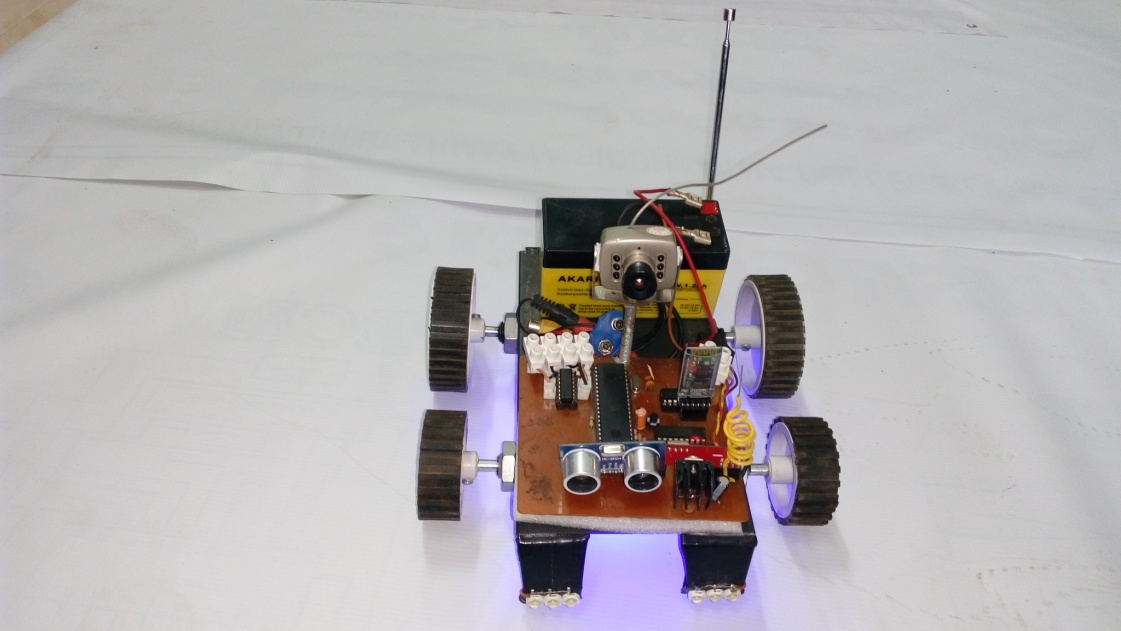
**TX\_Int(Distance);//Send distance to serial**

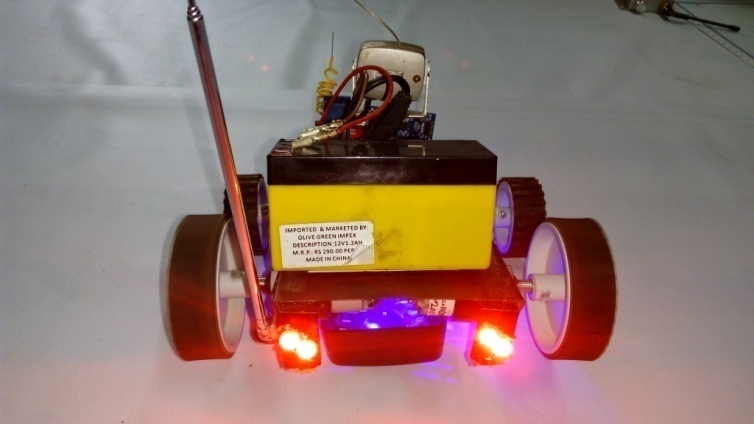
**TX\_String("\r\n");**

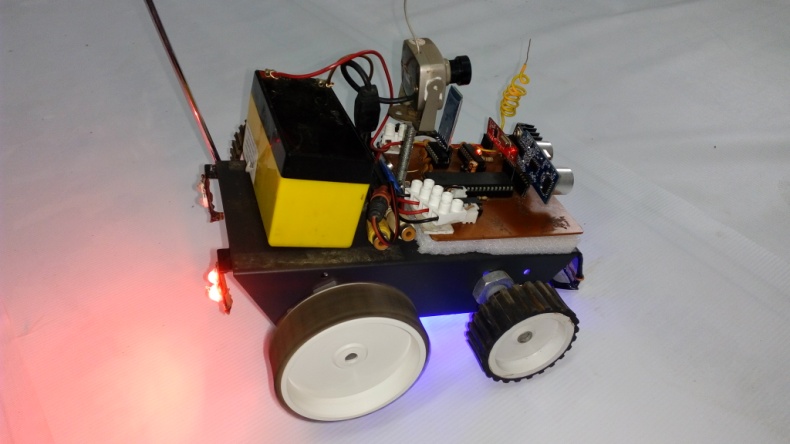
**DelayMs(2000);**

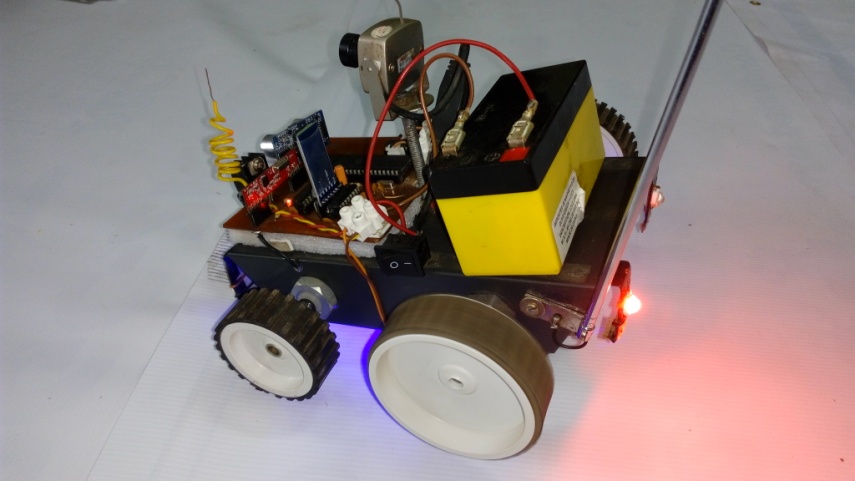
**}**

****

****

****

****

****