MOTHER BOARD (At-89c2051)

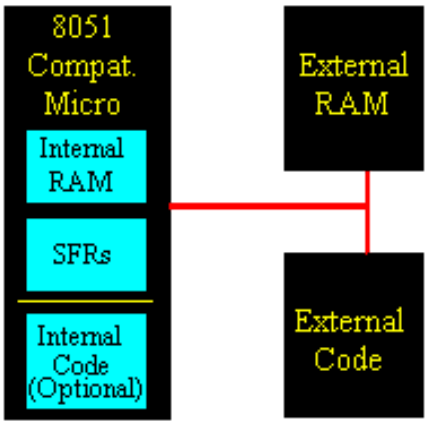
The motherboard of this project is designed with a MSC –51 core compatible micro controller. The motherboard is designed on a printed circuit board, compatible for the micro controller. This board is consisting of a socket for micro controller, input /output pull-up registers; oscillator section and auto reset circuit.

Micro controller core processor:

Introduction

The AT89C2051 is a low-voltage, high-performance CMOS 8-bit microcomputer with 2K bytes of Flash programmable and erasable read-only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard MCS-51 instruction set. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C2051 is a power-ful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications. The AT89C2051 provides the following standard features: 2K bytes of Flash, 128 bytes of RAM, 15 I/O lines, two 16-bit

timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, a precision analog comparator, on-chip oscillator and clock circuitry. In addition, the AT89C2051 is designed with static logic for opera-tion down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset The memory types are illustrated in the following graphic. They are: On-Chip Memory, External Code Memory, and External RAM.



On-Chip Memory refers to any memory (Code, RAM, or other) that physically exists on the Microcontroller itself. On-chip memory can be of several types, but we'll get into that shortly.

External Code Memory is code (or program) memory that resides off-chip. This is often in the form of an external EPROM.

External RAM is RAM memory that resides off-chip. This is often in the form of standard static RAM or flash RAM.

Code Memory

Code memory is the memory that holds the actual 8051 program that is to be run. This memory is limited to 64K and comes in many shapes and sizes: Code memory may be found *on-chip*, either burned into the Microcontroller as ROM or EPROM. Code may also be stored completely *off-chip* in an external ROM or, more commonly, an external EPROM. Flash RAM is also another popular method of storing a program. Various combinations of these memory types may also be used--that is to say, it is possible to have 4K of code memory *on-chip* and 64k of code memory *off-chip* in an EPROM.

When the program is stored on-chip the 64K maximum is often reduced to 4k, 8k, or 16k. This varies depending on the version of the chip that is being used. Each version offers specific capabilities and one of the distinguishing factors from chip to chip is how much ROM/EPROM space the chip has.

However, code memory is most commonly implemented as off-chip EPROM. This is especially true in low-cost development systems and in systems developed by students.

Programming Tip: Since code memory is restricted to 64K, 89C51 programs are limited to 64K. Some assemblers and compilers offer ways to get around this limit when used with specially wired hardware. However, without such special compilers and hardware, programs are limited to 64K.

External RAM

As an obvious opposite of *Internal RAM*, the 89C51 also supports what is called *External RAM*.

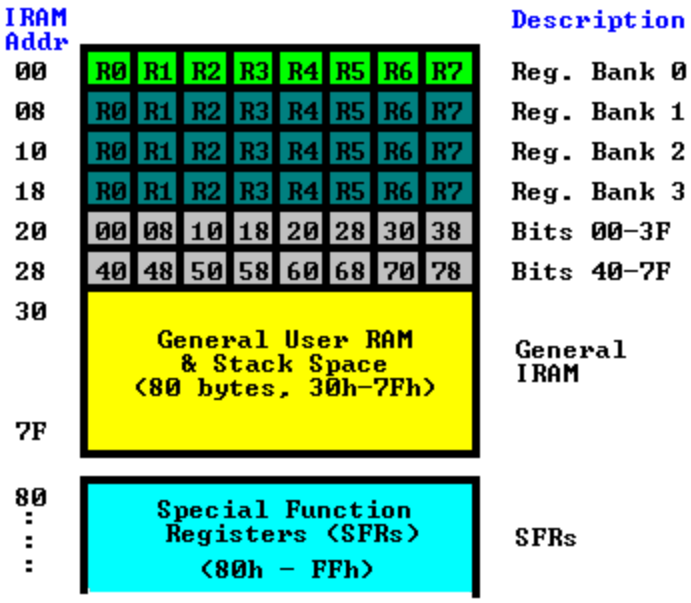
As the name suggests, External RAM is any random access memory which is found *off-chip*. Since the memory is off-chip it is not as flexible in terms of accessing, and is also slower. For example, to increment an Internal RAM location by 1 requires only 1 instruction and 1 instruction cycle. To increment a 1-byte value stored in External RAM requires 4 instructions and 7 instruction cycles. In this case, external memory is 7 times slower!

What External RAM loses in speed and flexibility it gains in quantity. While Internal RAM is limited to 128 bytes (256 bytes with an 8052), the 8051 supports External RAM up to 64K.

Programming Tip: The 8051 may only address 64k of RAM. To expand RAM beyond this limit requires programming and hardware tricks. You may have to do this "by hand" since many compilers and assemblers, while providing support for programs in excess of 64k, do not support more than 64k of RAM. This is rather strange since it has been my experience that programs can usually fit in 64k but often RAM is what is lacking. Thus if you need more than 64k of RAM, check to see if your compiler supports it-- but if it doesn't, be prepared to do it by hand.

On-Chip Memory

As mentioned at the beginning of this chapter, the 89C51 includes a certain amount of on-chip memory. On-chip memory is really one of two types: Internal RAM and Special Function Register (SFR) memory. The layout of the 89C51's internal memory is presented in the following memory map:



As is illustrated in this map, the 8051 has a bank of 128 bytes of *Internal RAM*. This Internal RAM is found *on-chip* on the 8051 so it is the fastest RAM available, and it is also the most flexible in terms of reading, writing, and modifying it's contents. Internal RAM is volatile, so when the 8051 is reset this memory is cleared.

The 128 bytes of internal ram is subdivided as shown on the memory map. The first 8 bytes (00h - 07h) are "register bank 0". By manipulating certain SFRs, a program may choose to use register banks 1, 2, or 3. These alternative register banks are located in internal RAM in addresses 08h through 1Fh. We'll discuss "register banks" more in a later chapter. For now it is sufficient to know that they "live" and are part of internal RAM.

Bit Memory also lives and is part of internal RAM. We'll talk more about bit memory very shortly, but for now just keep in mind that bit memory actually resides in internal RAM, from addresses 20h through 2Fh.

The 80 bytes remaining of Internal RAM, from addresses 30h through 7Fh, may be used by user variables that need to be accessed frequently or at high-speed. The Microcontroller as a storage area for the operating stack also utilizes this area. This fact severely limits the 8051's stack since, as illustrated in the memory map, the area reserved for the stack is only 80 bytes--and usually it is less since this 80 bytes has to be shared between the stack and user variables.

SFR Descriptions

There are different special function registers (SFR) designed in side the 89C51 micro controller. In this micro controller all the input , output ports, timers interrupts are controlled by the SFRs. The SFR functionalities are as follows.

This section will endeavor to quickly overview each of the standard SFRs found in the above SFR chart map. It is not the intention of this section to fully explain the functionality of each SFR--this

information will be covered in separate chapters of the tutorial. This section is to just give you a general idea of what each SFR does.

P0 (Port 0, Address 80h, Bit-Addressable): This is input/output port 0. Each bit of this SFR corresponds to one of the pins on the Microcontroller. For example, bit 0 of port 0 is pin P0.0, bit 7 is pin P0.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

Programming Tip: While the 8051 has four I/O port (P0, P1, P2, and P3), if your hardware uses external RAM or external code memory (i.e., your program is stored in an external ROM or EPROM chip or if you are using external RAM chips) you may not use P0 or P2. This is because the 8051 uses ports P0 and P2 to address the external memory. Thus if you are using external RAM or code memory you may only use ports P1 and P3 for your own use.

SP (Stack Pointer, Address 81h): This is the stack pointer of the Microcontroller. This SFR indicates where the next value to be taken from the stack will be read from in Internal RAM. If you push a value

onto the stack, the value will be written to the address of SP + 1. That is to say, if SP holds the value 07h, a PUSH instruction will push the value onto the stack at address 08h. This SFR is modified by all instructions which modify the stack, such as PUSH, POP, LCALL, RET, RETI, and whenever interrupts are provoked by the Microcontroller.

Programming Tip: The SP SFR, on startup, is initialized to 07h. This means the stack will start at 08h and start expanding upward in internal RAM. Since alternate register banks 1, 2, and 3 as well as the user bit variables occupy internal RAM from addresses 08h through 2Fh, it is necessary to initialize SP in your program to some other value if you will be using the alternate register banks and/or bit memory. It's not a bad idea to initialize SP to 2Fh as the first instruction of every one of your programs unless you are 100% sure you will not be using the register banks and bit variables.

DPL/DPH (Data Pointer Low/High, Addresses 82h/83h): The SFRs DPL and DPH work together to represent a 16-bit value called the *Data Pointer*. The data pointer is used in operations regarding external RAM and some instructions involving code

memory. Since it is an unsigned two-byte integer value, it can represent values from 0000h to FFFFh (0 through 65,535 decimal).

Programming Tip: DPTR is really DPH and DPL taken together as a 16-bit value. In reality, you almost always have to deal with DPTR one byte at a time. For example, to push DPTR onto the stack you must first push DPL and then DPH. You can't simply plush DPTR onto the stack. Additionally, there is an instruction to "increment DPTR." When you execute this instruction, the two bytes are operated upon as a 16-bit value. However, there is no instruction that decrements DPTR. If you wish to decrement the value of DPTR, you must write your own code to do so.

PCON (Power Control, Addresses 87h): The Power Control SFR is used to control the 8051's power control modes. Certain operation modes of the 8051 allow the 8051 to go into a type of "sleep" mode, which requires much, less power. These modes of operation are controlled through PCON. Additionally, one of the bits in PCON is used to double the effective baud rate of the 8051's serial port.

TCON (Timer Control, Addresses 88h, Bit-Addressable): The Timer Control SFR is used to configure and modify the way in which

the 8051's two timers operate. This SFR controls whether each of the two timers is running or stopped and contains a flag to indicate that each timer has overflowed. Additionally, some non-timer related bits are located in the TCON SFR. These bits are used to configure the way in which the external interrupts are activated and also contain the external interrupt flags which are set when an external interrupt has occurred.

TMOD (Timer Mode, Addresses 89h): The Timer Mode SFR is used to configure the mode of operation of each of the two timers. Using this SFR your program may configure each timer to be a 16-bit timer, an 8-bit auto reload timer, a 13-bit timer, or two separate timers. Additionally, you may configure the timers to only count when an external pin is activated or to count "events" that are indicated on an external pin.

TL0/TH0 (Timer 0 Low/High, Addresses 8Ah/8Ch): These two SFRs, taken together, represent timer 0. Their exact behavior depends on how the timer is configured in the TMOD SFR; however, these timers always count up. What is configurable is how and when they increment in value.

TL1/TH1 (Timer 1 Low/High, Addresses 8Bh/8Dh): These two SFRs, taken together, represent timer 1. Their exact behavior depends on how the timer is configured in the TMOD SFR; however, these timers always count up. What is configurable is how and when they increment in value.

P1 (Port 1, Address 90h, Bit-Addressable): This is input/output port 1. Each bit of this SFR corresponds to one of the pins on the Microcontroller. For example, bit 0 of port 1 is pin P1.0, bit 7 is pin P1.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

SCON (Serial Control, Addresses 98h, Bit-Addressable): The Serial Control SFR is used to configure the behavior of the 8051's on-board serial port. This SFR controls the baud rate of the serial port, whether the serial port is activated to receive data, and also contains flags that are set when a byte is successfully sent or received.

Programming Tip: To use the 8051's on-board serial port, it is generally necessary to initialize the following SFRs: SCON, TCON, and TMOD. This is because SCON controls the serial port. However,

in most cases the program will wish to use one of the timers to establish the serial port's baud rate. In this case, it is necessary to configure timer 1 by initializing TCON and TMOD.

SBUF (Serial Control, Addresses 99h): The Serial Buffer SFR is used to send and receive data via the on-board serial port. Any value written to SBUF will be sent out the serial port's TXD pin. Likewise, any value which the 8051 receives via the serial port's RXD pin will be delivered to the user program via SBUF. In other words, SBUF serves as the output port when written to and as an input port when read from.

P2 (Port 2, Address Aoh, Bit-Addressable): This is input/output port 2. Each bit of this SFR corresponds to one of the pins on the Microcontroller. For example, bit 0 of port 2 is pin P2.0, bit 7 is pin P2.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

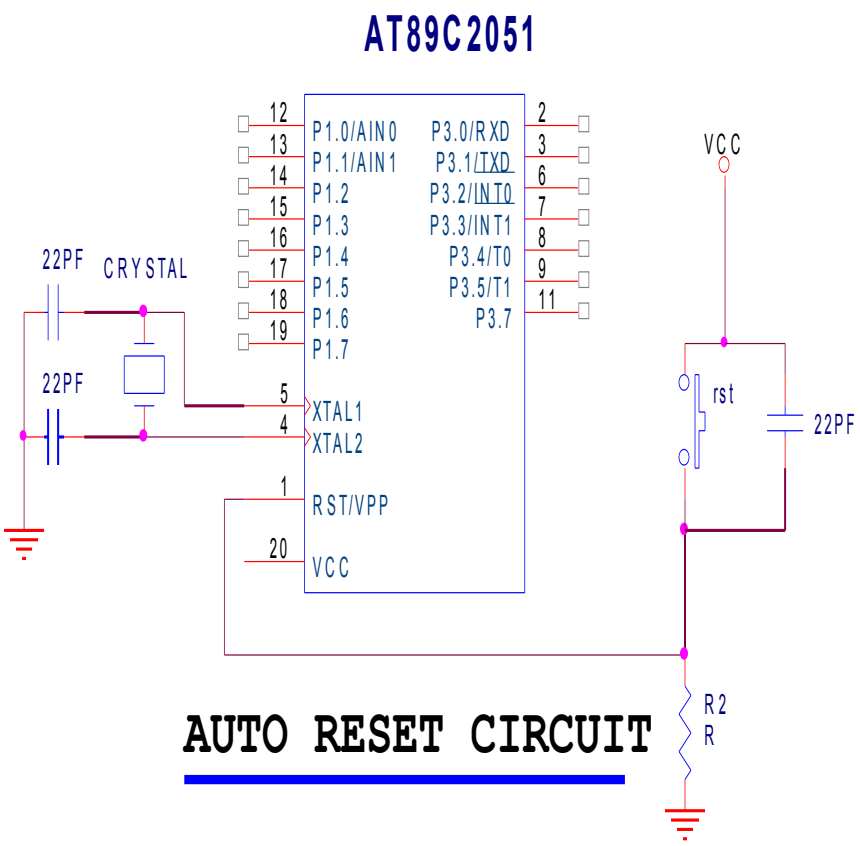
Programming Tip: While the 8051 has four I/O port (P0, P1, P2, and P3), if your hardware uses external RAM or external code

memory (i.e., your program is stored in an external ROM or EPROM chip or if you are using external RAM chips) you may not use P0 or P2. This is because the 8051 uses ports P0 and P2 to address the external memory. Thus if you are using external RAM or code memory you may only use ports P1 and P3 for your own use.

IE (Interrupt Enable, Addresses A8h): The Interrupt Enable SFR is used to enable and disable specific interrupts. The low 7 bits of the SFR are used to enable/disable the specific interrupts, where as the highest bit is used to enable or disable ALL interrupts. Thus, if the high bit of IE is 0 all interrupts are disabled regardless of whether an individual interrupt is enabled by setting a lower bit.

P3 (Port 3, Address B0h, Bit-Addressable): This is input/output port 3. Each bit of this SFR corresponds to one of the pins on the Micro controller. For example, bit 0 of port 3 is pin P3.0, bit 7 is pin P3.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

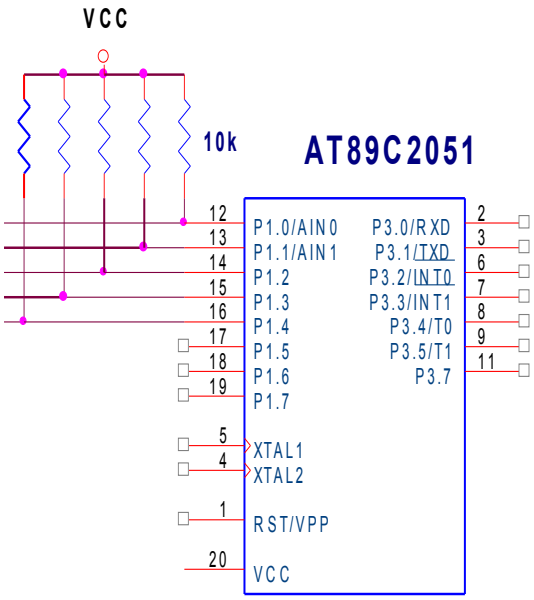
Auto reset Circuit:



The auto reset circuit is a RC network as shown in the mother board circuit diagram. A capacitor of 1-10mfd is connected in series with a 8k2 resister the R-C junction is connected to the micro controller pin –9 which is reset pin. The reset pin is one when ever kept high(logic

1) the programme counter (PC) content resets to 0000h so the processor starts executing the programme. from that location. When ever the system is switched ON the mother board gets power and the capacitor acts as short circuit and the entire voltage appears across the resistor, so the reset pin get a logic 1 and the system get reset, whenever it is being switched ON.

Pull-UP Resisters:

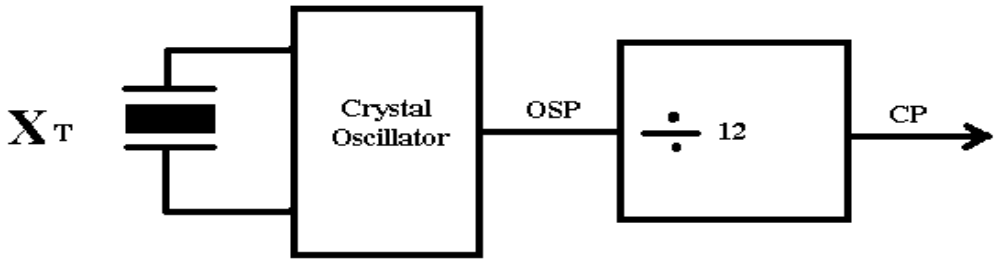


PULL UP RESISTORS

The PORT1 and PORT3 of the MCS-51 architecture is of open collector type so on writing logic 0 the pins are providing a perfect ground potential. Where as on writing logic 1 the port pins behaves as high impedance condition so putting a pull-up resister enables the port to provide a +5volt(logic 1). Port1 and Port3 are provided with internal pull-ups. A pull-up resister is normally a 10K resistance connected from the port pin to the Vcc (+5) volt.

Crystal Oscillator

The 8051 family microcontroller contains an inbuilt crystal oscillator, but the crystal has to be connected externally. This family of microcontroller can support 0 to 24MHz crystal and two numbers of decoupling capacitors are connected as shown in the figure. These capacitors are decouples the charges developed on the crystal surface due to piezoelectric effect. These decoupling capacitors are normally between 20pf to 30pf. The clock generator section is designed as follows,



12 OSP = CP

The Microcontroller design consist of two parts

1) Hardware.
2) Software.

HARDWARE: The controller operates on +5 V dc, so the regulated + 5v is supplied to pin no. 40 and ground at pin no. 20. The controller is used here need not required to handle high frequency signals, so as 4 MHz crystal is used for operating the processor. The pin no. 9 is supplied with a +5V dc through a push switch. To reset the processor

.As prepare codes are store in the internal flash memory the pin no. 31 is connected to + Vcc

Proximity sensor

Detecting Obstacle with IR (Infrared) Sensor

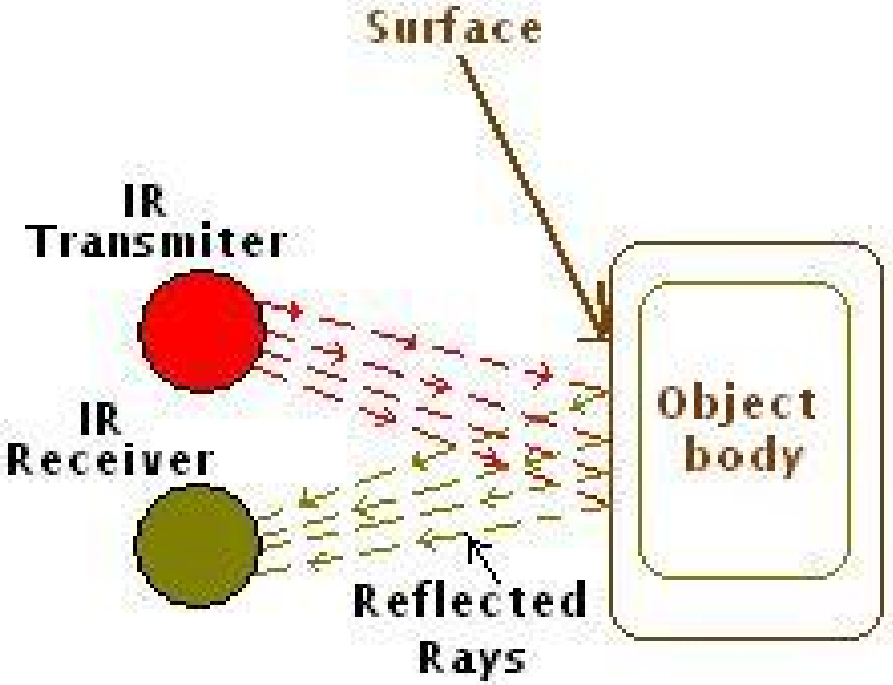
The basic concept of IR (infrared) obstacle detection is to transmit the IR signal(radiation) in a direction and a signal is received at the IR receiver when the IR radiation bounces back from a surface of the object.

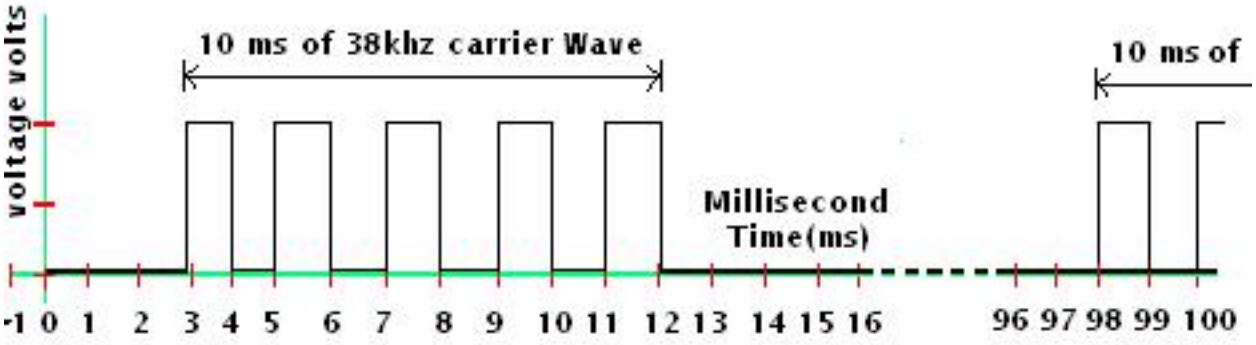
Here in the figure the object can be anything which has certain shape and size, the IR LED transmits the IR signal on to the object and the signal is reflected back from the surface of the object. The reflected signals are received by an IR receiver. The IR receiver can be a photodiode / phototransistor or a readymade module which decodes the signal.

In order to implement the IR obstacle detection, we need to understand the following

- We need to understand how to transmit IR signal using commercially available electronic components.
- Same way we also need to understand the IR receiver.

My main focus in this document is to explain the implementation of IR based obstacle detection in detail.





IR Transmitter

In general, the basic building block of any IR transmitter is modulation of the information signal with carrier signal, because the receiver modules which are available off-the-shelf are made for a particular carrier frequency. So it is clear that when you chose a particular IR receiver module, you also need to transmit the the

modulated wave with the same carrier frequency of that of a IR receiver module.

Modulating a 38 Khz carrier signal

ON state = 10ms

OFF state = 90ms

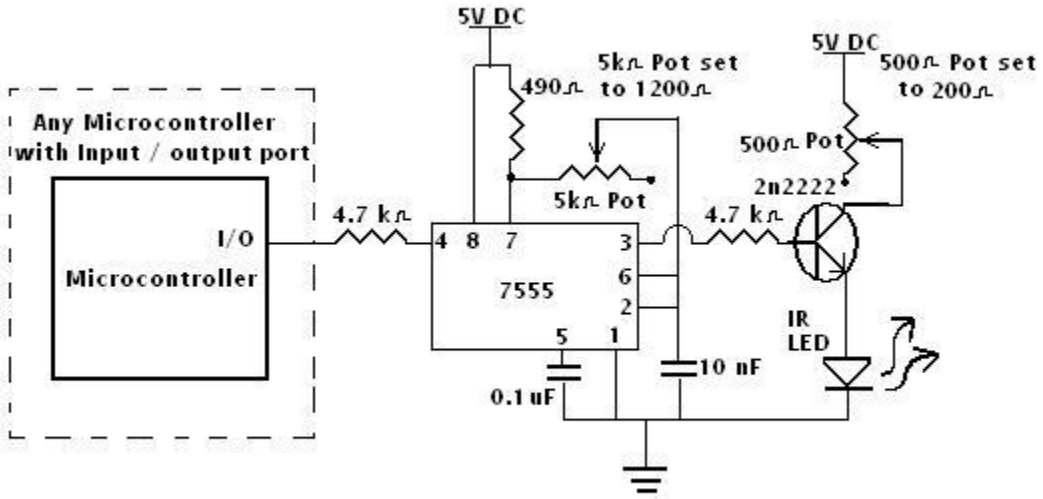
The figure above explains the modulation process, this is similar to OOK(ON-OFF Keying) modulation, where the carrier signal is ON for certain period of time. When transmitting a signal for obstacle detection, it is necessary that the carrier signal is transmitted for a short while and remains OFF for longer period of time.

If the transmission of the carrier signal is prolonged, in other words, instead of having a short transmission period(10 milliseconds in our case, as explained in the figure) of carrier signal, if we have it for a long period of time then the receiver module will treat it as a noise and ignores receiving the transmitted signal.

The implementation of IR transmitter can be done in various ways; in this document we will discuss two ways

- Using 7555(compatible with 555) timer IC to generate a 38 kHz carrier signal
- Using Micro controller(Atmel atmega8535) inbuilt wave generation module

Now let us have a look at the IR transmitter using 7555 timer IC

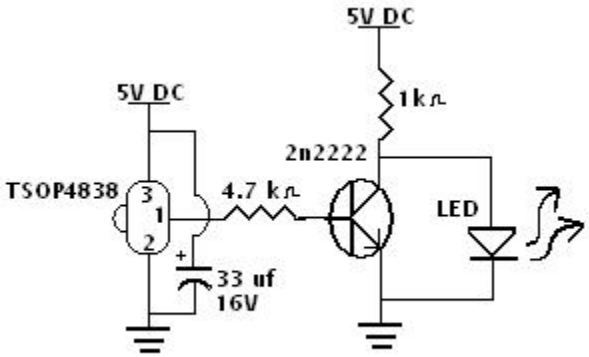


IR Receiver

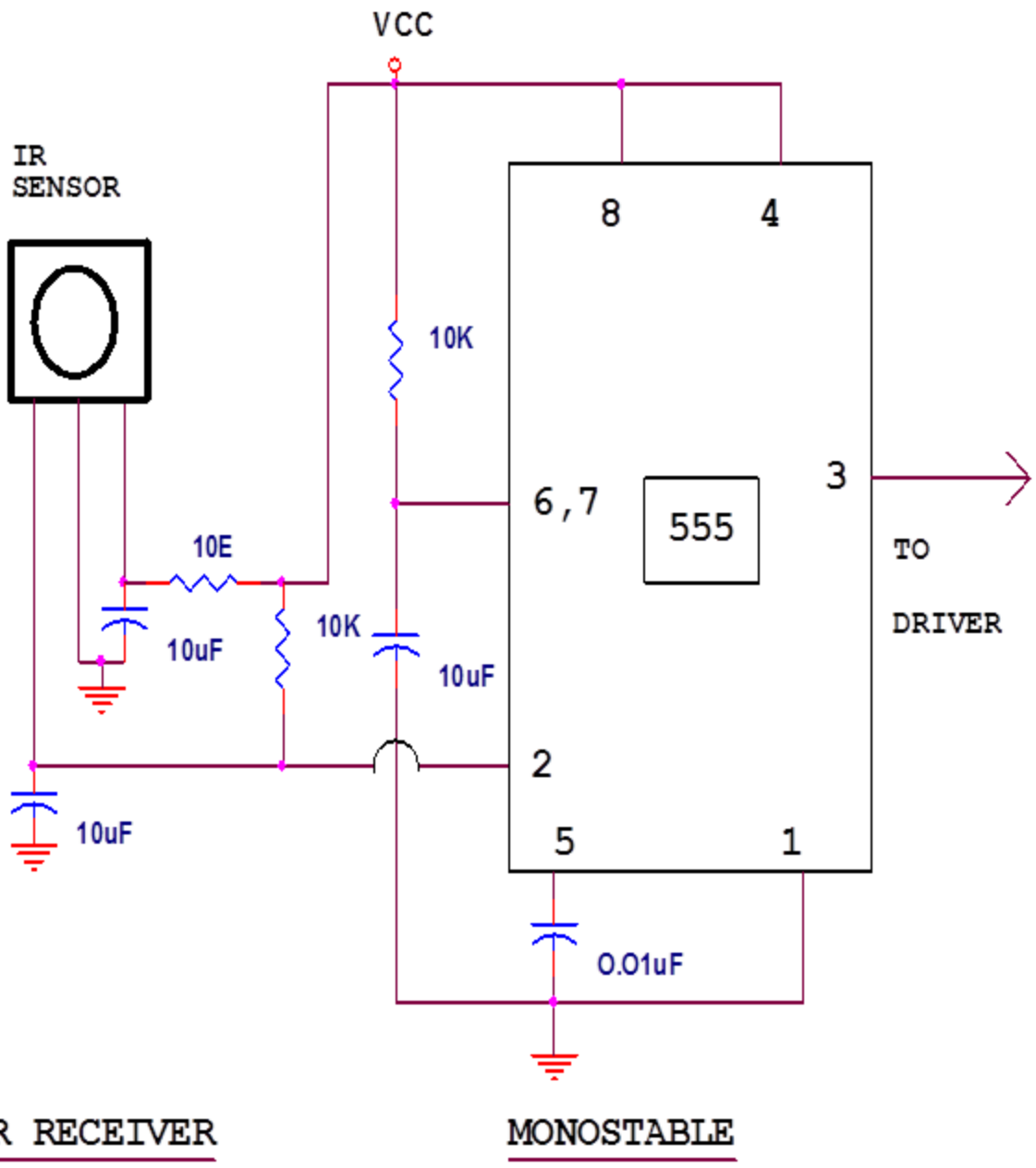
It is quite simple to construct a IR receiver with readily available off-the-shelf modules. These modules are nothing but the IC packages, referred as TSOP (Thin small-outline package). In this document, the receiver is designed for 38 kHz carrier signal; hence the IC selected should work for the same frequency. The IC TSOP4838 will serve as a receiver module, which is compatible with both TTL and CMOS logic. This means that we can directly get digital signal from the receiver module and then connect it to the microcontroller.

The Implementation of IR receiver is explained using an LED as an indicator.

Here in the circuit the LED blinks whenever the TSOP4838 module receives a signal from the transmitter. The same circuit can be altered to work with microcontroller; the circuit below has both IR transmitter and IR receiver modules integrated with the microcontroller.



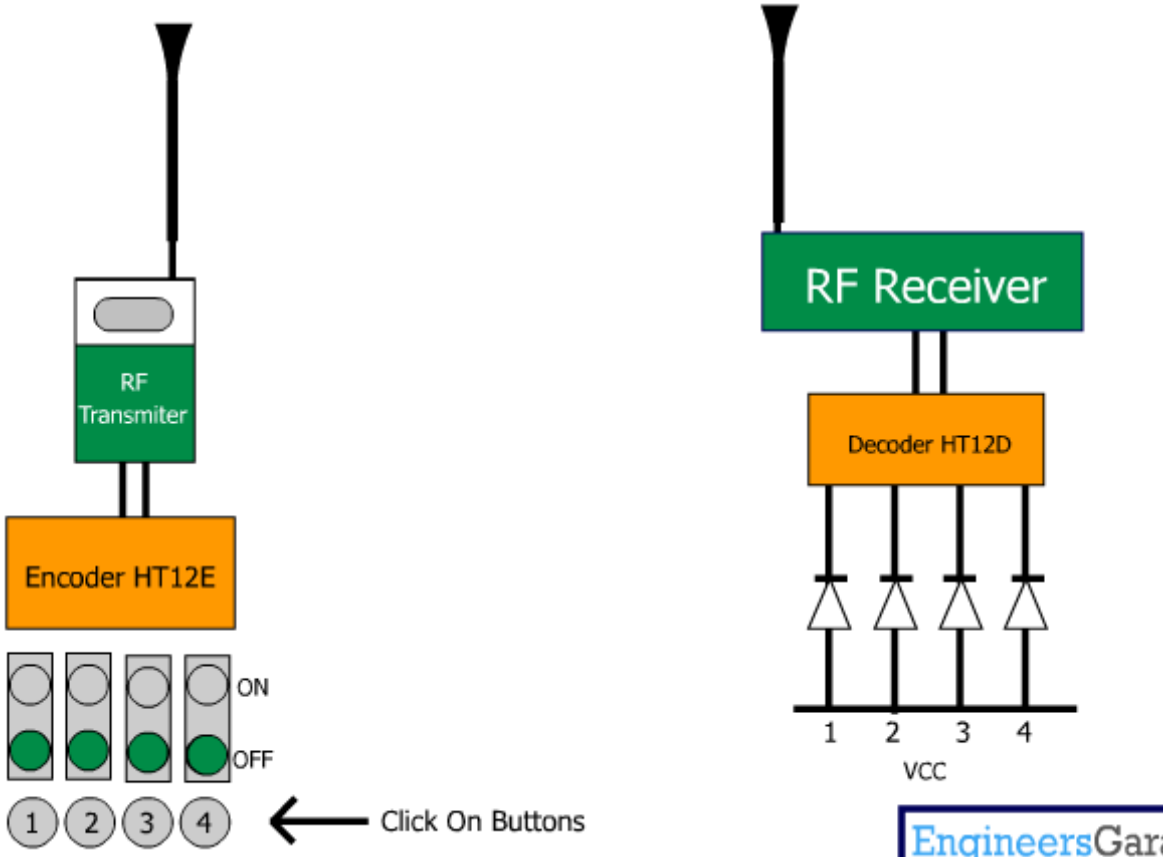
Diagram



RF transmitter and receiver

This circuit utilizes the RF module (Tx/Rx) for making a wireless remote, which could be used to drive an output from a distant place.

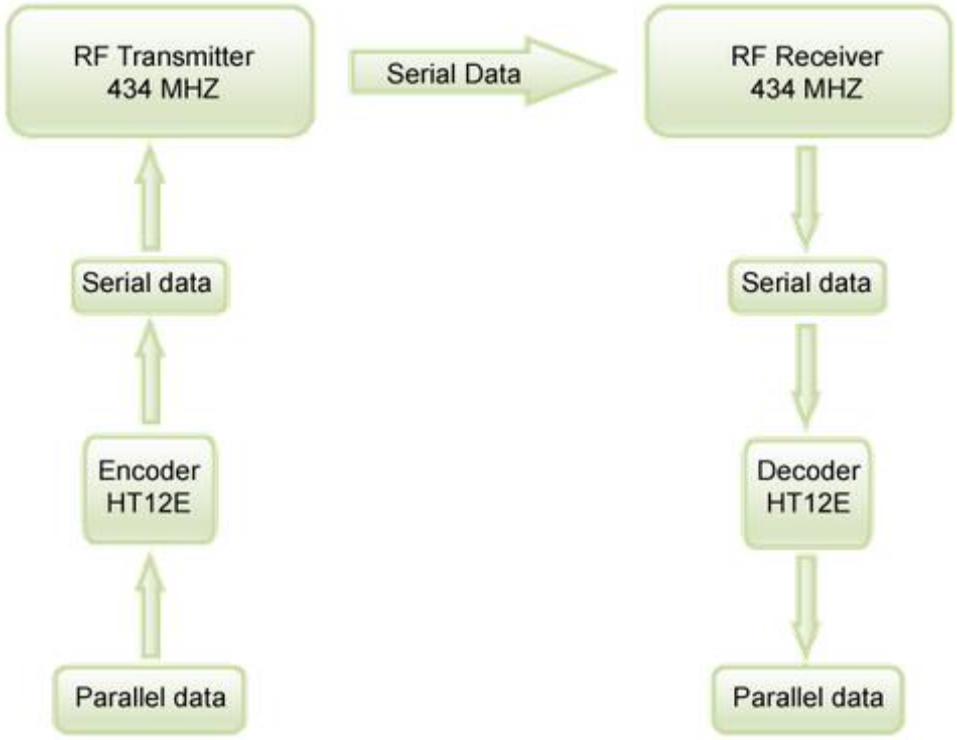
RF module, as the name suggests, uses radio frequency to send signals. These signals are transmitted at a particular frequency and a baud rate. A receiver can receive these signals only if it is configured for that frequency. A four channel encoder/decoder pair has also been used in this system. The input signals, at the transmitter side, are taken through four switches while the outputs are monitored on a set of four LEDs corresponding to each input switch. The circuit can be used for designing Remote Appliance Control system. The outputs from the receiver can drive corresponding relays connected to any household appliance.



Description - This radio frequency (RF) transmission system employs Amplitude Shift Keying (ASK) with transmitter/receiver (Tx/Rx) pair operating at 434 MHz. The transmitter module takes serial input and transmits these signals through RF. The transmitted signals are received by the receiver module placed away from the source of transmission. The system allows one way communication between two nodes, namely, transmission and reception. The RF module has been used in conjunction with a set of four channel encoder/decoder ICs. Here HT12E & HT12D have been used as encoder and decoder respectively. The encoder converts the parallel inputs (from the remote switches) into serial set of signals. These signals are serially transferred through RF to the reception point. The decoder is used after the RF receiver to decode the serial format and retrieve the original signals as outputs. These outputs can be observed on corresponding LEDs.

Encoder.

Block



Encoder IC (HT12E) receives parallel data in the form of address bits and control bits. The control signals from remote switches along with 8 address bits constitute a set of 12 parallel signals. The encoder HT12E encodes these parallel signals into serial bits. Transmission is enabled by providing ground to pin14 which is active low. The control signals are given at pins 10-13 of HT12E. The serial data is fed to the RF transmitter through pin17 of HT12E.

Transmitter, upon receiving serial data from encoder IC (HT12E), transmits it wirelessly to the RF receiver. The receiver, upon receiving these signals, sends them to the decoder IC (HT12D) through pin2. The serial data is received at the data pin (DIN, pin14) of HT12D. The

decoder then retrieves the original parallel format from the received serial data.

When no signal is received at data pin of HT12D, it remains in standby mode and consumes very less current (less than 1μA) for a voltage of 5V. When signal is received by receiver, it is given to DIN pin (pin14) of HT12D. On reception of signal, oscillator of HT12D gets activated. IC HT12D then decodes the serial data and checks the address bits three times. If these bits match with the local address pins (pins 1-8) of HT12D, then it puts the data bits on its data pins (pins 10-13) and makes the VT pin high. An LED is connected to VT pin (pin17) of the decoder. This LED works as an indicator to indicate a valid transmission. The corresponding output is thus generated at the data pins of decoder IC. A signal is sent by lowering any or all the pins 10-13 of HT12E and corresponding signal is received at receiver's end (at HT12D). Address bits are configured by using the by using the first 8 pins of both encoder and decoder ICs. To send a particular signal, address bits must be same at encoder and decoder ICs. By configuring the address bits properly, a single RF transmitter can also be used to control different RF receivers of same frequency.

To summarize, on each transmission, 12 bits of data is transmitted consisting of 8 address bits and 4 data bits. The signal is received at receiver's end which is then fed into decoder IC. If address bits get matched, decoder converts it into parallel data and the corresponding data bits get lowered which could be then used to drive the LEDs. The outputs from this system can either be used in negative logic or NOT gates (like 74LS04) can be incorporated at data pins.

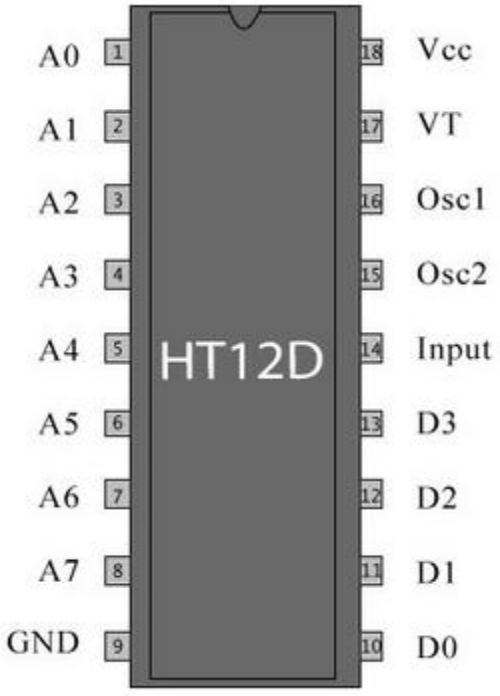
COMPONENTS USED

1. HT12D DECODER

Download Datasheet: HT12D.pdf

HT12D IC comes from HolTek Company. 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.

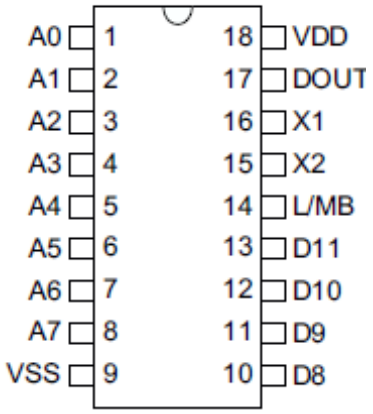


HT12E ENCODER

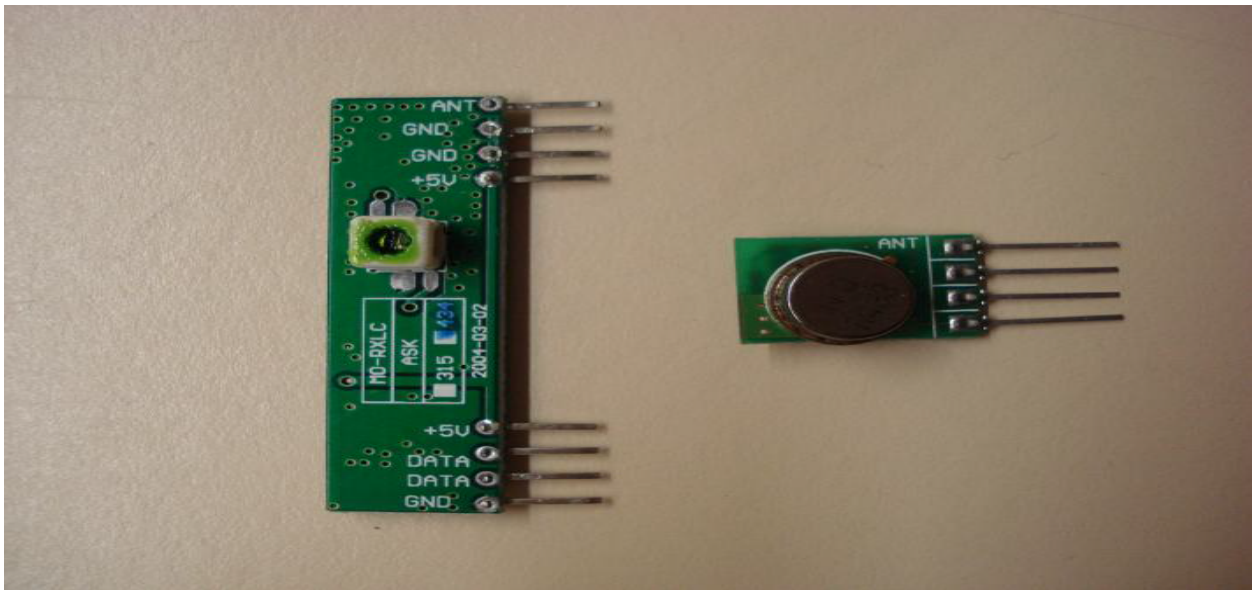
Download Datasheet: HT12E.pdf

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.

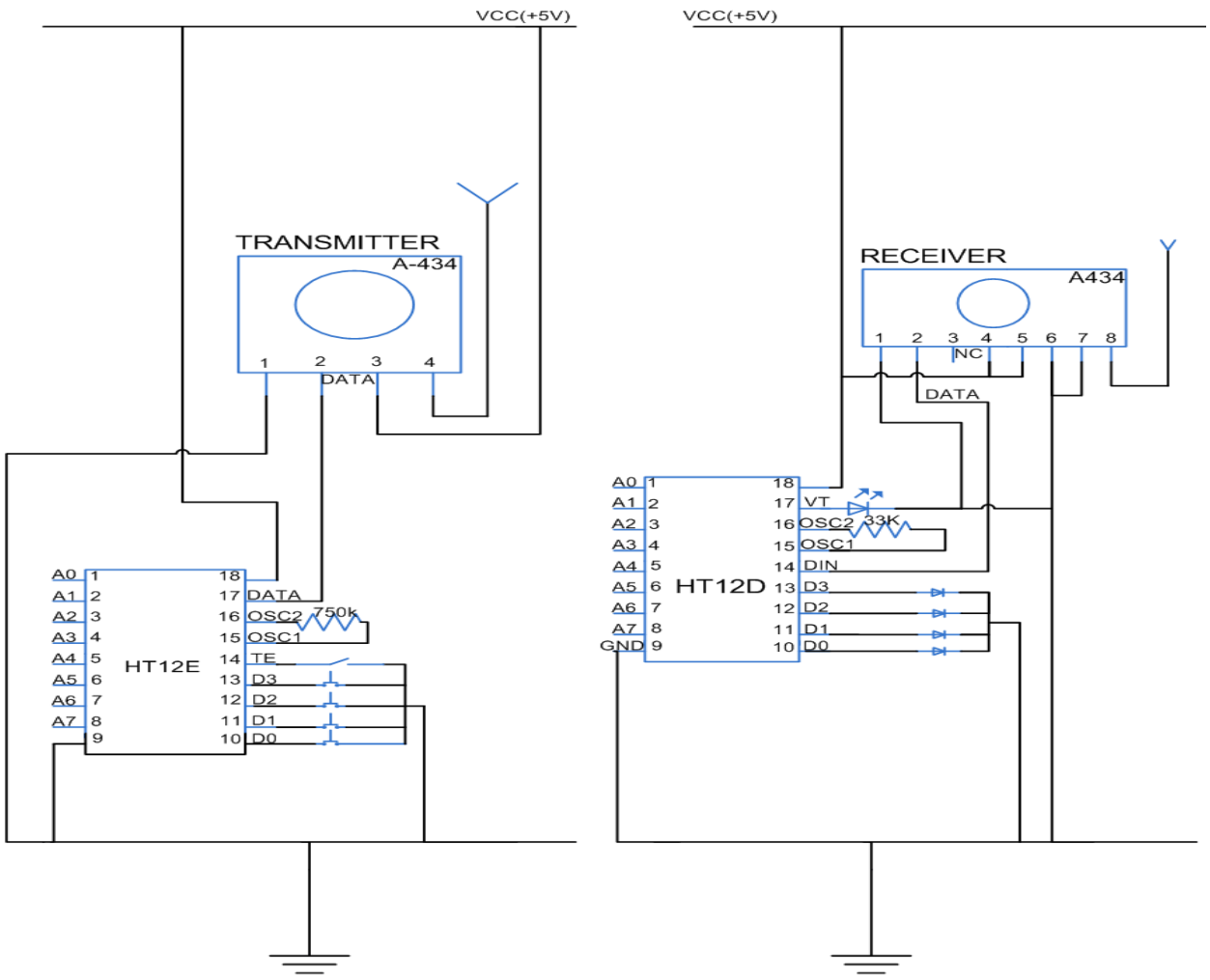
Pin Diagram Pin Description



Photo



circuit



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.

Motor Driver

The D.C. Motor used in this project operates at 12 volt and carries approximately 400mA of current. The motor driver is designed to

inter face the motor with micro controller. The micro controller output is +5volt and can maximum give a current of 5mA. The driver stage changes the current and voltage level suitably to drive the motor. The driver stage not only drives the motor but also helps to control the direction of rotation. As the output current (Ic) is large the driver section requires a Darlington pair to switch the load. The Darlington pair I.C. TIP 122 is used here for designing. There are four ICs used here but two of those switched for one direction and other two will be switched for opposite direction rotation of the D.C. motor. The design principle of the driver section is as follows.

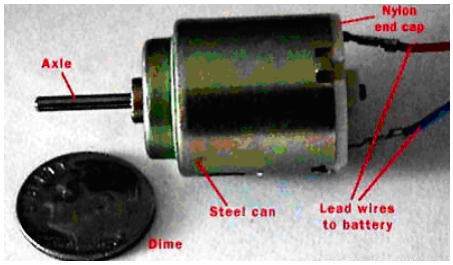
The motor takes approximately 400mA at 12 volt D.C., The power transistors can have amplification factor maximum 60 to 70 as per this assumption the base current required to switch on the transistor is approximately

$$Ib = (Ic/beta) = 400mA/60 = 6.7\ mA$$

This current is too high to supply as a base current, more over the Microcontroller can not supply that much current to drive the transistor so, a darling ton pair is required to limit the base current with in 100 micro amp. To 2 mA.

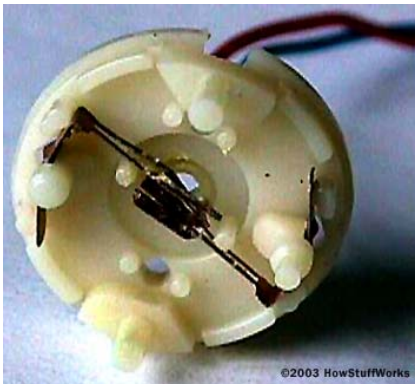
DC MOTOR

The motor being dissected here is a simple PMDC electric motor that is typically find applications in robotics and control systems also used for techo generator in the industries.



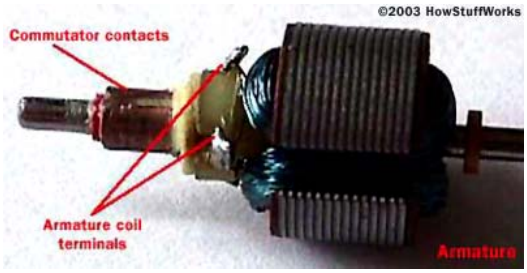
This is a small motor, about as big around as a coin. From the outside the body of the motor is shown in the picture along with its axle and two battery leads. If the motor is connected to the battery then , the axle will spin. If the leads are reversed then, it will spin in the opposite direction. Here are two other views of the same motor. (Note the two slots in the side of the steel can in the second shot -- their purpose will become more evident in a moment.)

e nylon end cap is held in place by two tabs that are part of the steel can. By bending the tabs back, end cap can be free and removed. Inside the end cap are the motor's brushes. These brushes transfer power from the battery to the commutator as the motor spins:



The axle holds the armature and the commutator. The armature is a set of electromagnets in this case three. The armature in this motor is a set of thin metal plates stacked together, with thin copper wire

coiled around each of the three poles of the armature. The two ends of each wire (one wire for each pole) are soldered onto a terminal, and then each of the three terminals is wired to one plate of the commutator. The figures below make it easy to see the armature, terminals and commutator:

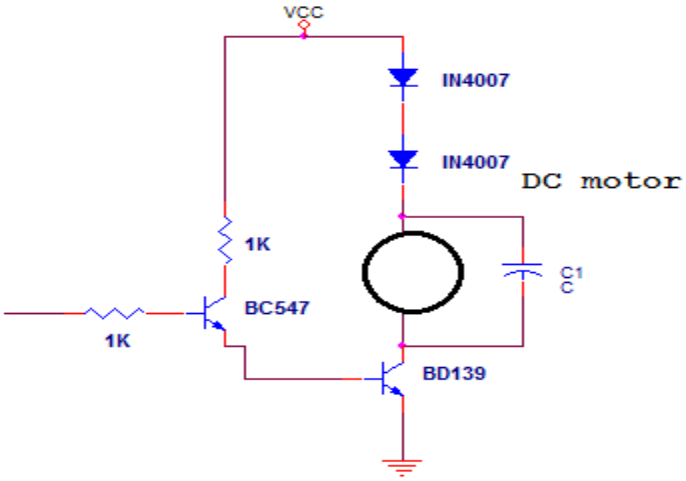


The final piece of any DC electric motor is the field magnet. The field magnet in this motor is formed by the can itself plus two curved permanent magnets:

One end of each magnet rests against a slot cut into the can, and then the retaining clip presses against the other ends of both magnets.

An electromagnet is the basis of an electric motor. You can understand how things work in the motor by imagining the following scenario. Say that you created a simple electromagnet by wrapping 100 loops of wire around a nail and connecting it to a battery. The nail would become a magnet and have a north and south pole while the battery is connected.

Now say that you take your nail electromagnet, run an axle through the middle of it and suspend it in the middle of a horseshoe magnet as shown in the figure below. If you were to attach a battery to the electromagnet so that the north end of the nail appeared as shown, the basic law of magnetism tells you what would happen: The north end of the electromagnet would be repelled from the north end of the horseshoe magnet and attracted to the south end of the horseshoe magnet. The south end of the electromagnet would be repelled in a similar way. The nail would move about half a turn and then stop in the position shown.



Code

```
$mod51
org 000h


mov p1, #0ffh

mov P3, #0ffh



nex3:   jb p1.0, nex3
acall delay_1s
jb p1.0, nex3
acall delay_1s
jb p1.0,nex3

acall buzzer
sjmp nex3
```

```
buzzer: clr p1.1        ;buzzer


acall delay
                        acall delay



setb  p1.1
acall delay_1s

ret

        delay:
                        mov r0,#125d
                l3:     mov r1, #60d
                l2:     mov r2, #120d
                l1:     djnz r2, l1
                        djnz r1, l2
                        djnz r0, l3
                        ret

delay_1s:
                using 2
                mov r0, #202d
        xa3:    mov r1, #118d
        xa2:    mov r2, #7d
        xa1:    djnz r2, xa1
                djnz r1, xa2
                djnz r0, xa3
                using 0
                ret


                        end
```