

Miniaturized Remote Light

Saurabh Kwatra Resolve



Introduction

I was recently contacted by a gentleman who was looking for a very small circuit that would contain an LED that could be turned on and off with some sort of remote control. At first I responded to him that this project didn't hold much interest for me and that I would not be interested in pursuing it. Before writing it off entirely I asked for more information on his idea, specifically the size requirements and what the end use was to be.

When he responded that it should be no larger than a Tylenol capsule that gave me even more reason to pass on it. As luck would have it, however, his intended end use got my attention. He is a teacher and wants to use these small devices to highlight items on a black board (or more likely a white board these days!) so that they would stand out for his students. Having been a classroom teacher and then a technology coordinator I have a soft spot for teachers with technology-related ideas. When I was technology coordinator for my school district I went to great lengths to make technology work for my colleagues. I began to feel the same way about this endeavor.

Design Considerations

There are a number of issues with trying to design a device that is as small as a Tylenol capsule.

- Finding a very small battery with sufficient power to work for a reasonable amount of time
- Identifying a way to connect the battery to the circuit
- Determining the type of remote control receiver that could be used for turning the LED on and off
- Designing a control circuit to make it all happen
- Writing software so that the unit behaves as envisioned
- Fabricating the unit so that it will fit into a small space

Batteries

The biggest problem is supplying power while meeting the space restrictions. Small batteries like those for hearing aids and wrist watches are the obvious candidates but they don't contain

very much energy and small battery enclosures to hold them are not readily available. The only battery operated circuits utilizing small batteries that I had experience with were powered by 3.7 volt 2032 batteries. These supply a good bit of power (more than 200 mAh) but are about the size of a quarter dollar, decidedly larger than a small capsule. The most promising size that I have found at a reasonable price is part #AG5 which will supply about 50 mAh. Zinc air batteries of similar sizes give much more life but suffer from the fact that once "opened" and exposed to air they expire in a few weeks. Alkaline cells seem to give the best power for the price. Unfortunately they only supply 1.5 volts per cell and we need at least 3 volts to power the circuit so two need to be placed in series.

Battery Connection

Since we are not starting out with a container into which the battery and circuitry will be housed we must find a way to get things connected together. With non-rechargeable batteries it is not practical to solder directly to the battery even though this is the best way to keep size to a minimum. I explored other options and settled on using small rare earth magnets to hold the batteries to the circuit. The problem that this presents is how does one connect wires to the magnets?

One can solder to many rare earth magnets but heat, even for a very brief time, can destroy the magnetic domains and render the magnets weak or useless. I experimented with a number of methods of connecting wires to magnets and settled on a two step process. First I stripped about 1/2" of insulation from each end of a red and a black stranded wire. I pressed the wire against the small magnet and added a drop of CA (super glue) - while holding the wire in place with the edge of a screwdriver I brushed the still wet CA with accelerator. CA accelerator is a chemical that caused super glue to set almost instantly. See:

<http://sinbadglue.com/shop/Itemdetail.asp?ID=1132> . Once the CA was dry I tested the connection with an ohm meter. 0.2 ohms.... not bad! I added a coat of 5 minute epoxy to give a bit of a strain relief. I attached the two magnets to either side of a 2032 and measured the voltage.... 3.7 volts, just what I had hoped for!

Using small magnets opens up lots of options for the final design. I also found that a third small magnet can be placed between the two 1.5 volt alkaline cells to join them in series.

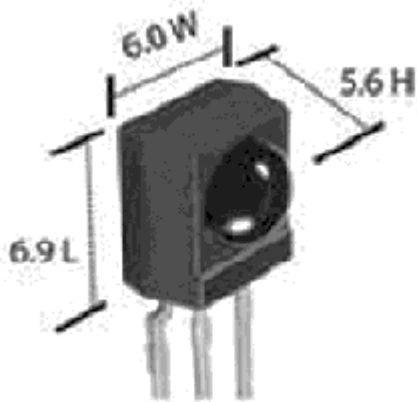
If you want to try your hand at soldering directly to the magnets make sure you use a very hot iron and get in and out in no more than a second or two. Immediately wet the magnet to cool it down. I found it beneficial to clamp the magnet into a vise or other fixture as the magnet will likely grab the iron tip on the soldering iron and it may be difficult to pull away if the magnet is not fixed to some other object.

Remote Control

There are two options that were considered for remote control. We could use either radio control or infrared. I felt that the infrared option was much simpler and likely to be less expensive and consume less power. I have done a good bit of work with using TV remote controls like this one to control various projects and have found them to be reliable and easy to use.



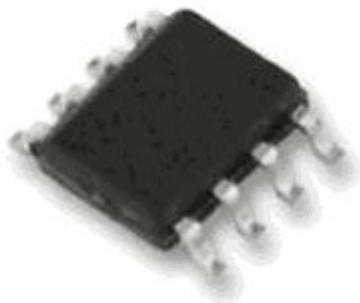
I also knew that the devices that are used to receive and interpret infrared signals from a TV remote are fairly small. The sensor, a [TSOP4038 from Vishay](#), is easily small enough at approximately 6mm on each side. It also works on as little as 2.7 volts so it will operate from two 1.5 volt cells connected in series. It will, however, cease working as the batteries discharge. The good news is that I have tested these devices with voltages as low as 2.1 volts and they continue to perform properly.



In addition the use of a TV remote brings the potential for much more customization of the devices.

Microcontroller Control

To pull it all together we need some sort of circuitry that will receive and act upon infrared pulses and turn the LED on or off as needed. While this could be done with discrete devices like transistors, resistors and capacitors it make much more sense to use a very small microcontroller to perform these tasks.



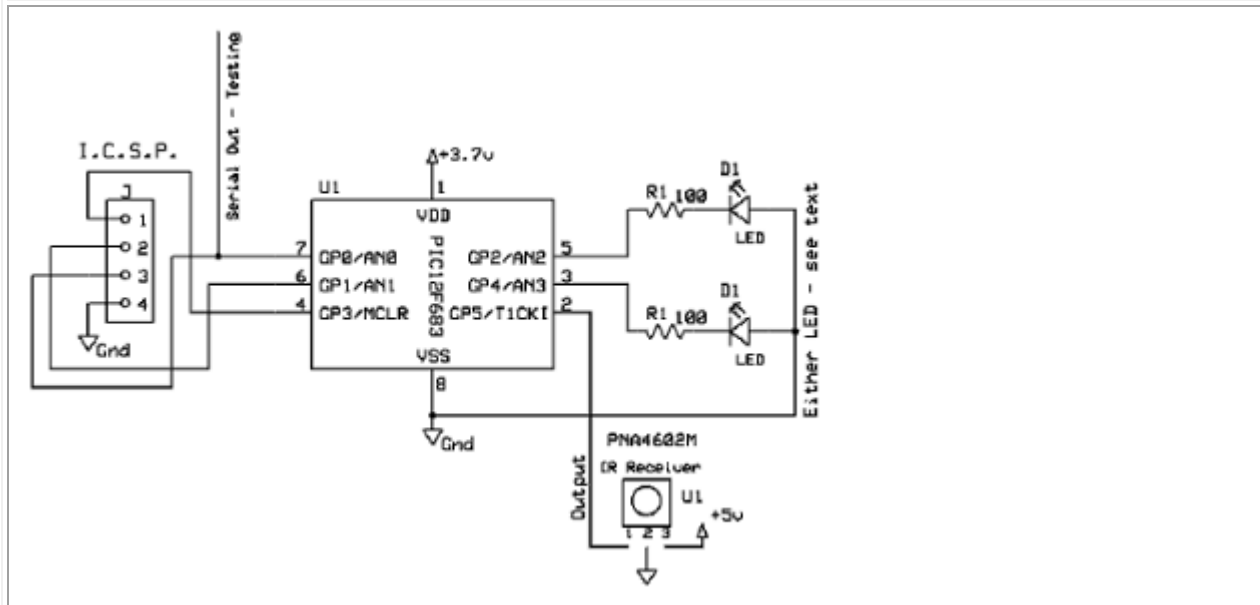
The [PIC 12F683](#) is an 8 pin device. I chose to use the surface mount version, shown above,

which measures about 5mm x 5mm x 2mm. It will operate on as little as 2.0 volts, a real plus in this application as it will continue to operate even when the batteries are nearly discharged.

As you can see in the schematic below two of the chip's pins (#1 and #8) are used for power and ground while two others (#2 and #3) connect to the LED and IR Receiver. The connections to pins 4, 6 and 7 are only used for programming and debugging and are not needed for the unit once it is complete.

The current limiting resistor, R1, is 100 ohms and keeps the amount of power that the LED pulls from the batteries to a minimum. A larger value resistor will extend battery life but will light the LED less brightly.

Note that two LEDs and current limiting resistors are shown. You need only install one of them. I included two as it might be easier to wire to one pin than the other. The software will activate both pins 3 and 5 when needed.



Software

The program that resides on the 12F683 is shown below. It is written in PIC Basic and compiled before it is sent to the microcontroller. It does the following:

- When first programmed each unit is given a number between 1 and 10. To simplify things each is first named "2".
- When the IR sensor sees its number the LED goes on and stays on
- The second time it sees its number it flashes at about 1 Hz (once per second) - note that this can increase in speed if exposed to fluorescent lights
- The third time it sees its number it goes dark.
- If the "MENU" key on the remote control is pressed *and the LED is in flash mode* the LED flashes its number (i.e. flashes once for "1", twice for "2", etc) - NOTE - MENU will NOT work if the LED is not flashing. This is designed to allow you to work near a number of these units and not change them all at once!

- The unit waits until a number from 1 to 10 (zero button) is pressed. It assigns that number to the unit so that it will only react to its number when pressed on the TV remote
- That number is written to internal memory so that it will not be lost when power is removed.
- A unit's number can be revealed by pressing the RETURN button. The LED will flash once for "1", twice for "2" and so on
- The device can be put into a very low power "Sleep" mode by pressing the POWER button. The LED will go from bright to dim indicating it is going to sleep. Press the POWER key again to wake it up

Giving each unit a unique number allows up to 10 of these devices to be used at the same time.

```
'd. bodnar 2-01-2012  Black Board Blinker
' CHANGES:
' DONE hit RETURN to show unit value - always ends in not flashing
' DONE press unit # once for solid on, a second time for flashing and a
third time for off
' DONE consider change so that MENU is only recognized if unit is flashing
(easier...
'     when working with multiple units)
' DONE add routine from Bike Blinker that puts it to sleep when POWER is
pressed
'     may need to move LED from pin 3 to pin 5 for PWM dimming /
brightening
' FIXED does not come back to mode (solid on) when it wakes up - does do so
on flash
' using both pins 3 and 5 for LED - added current limiting resistor, too
DEFINE PULSIN_MAX 10000
INTCON = %00001100      '00xx11xx to be CCP1 mode
ansel = 0               'all inputs digital
cmcon0 = 7
INCLUDE "modedefs.bas"
Serial_out      VAR gpio.0      'pin 7
irin           VAR gpio.5      'pin 2

NotUsed1       VAR gpio.1      'pin 6
LED            VAR gpio.4      'pin 3  NOTE - must use INTOSIO in
programmer

LED2           VAR gpio.2      'pin 5
trisiko       = %00100000      '
gpio          = %00100000

one            CON 128
two           CON 129
three        CON 130
four         CON 131
five        CON 132
six         CON 133
seven       CON 134
eight      CON 135
nine       CON 136
zero      CON 137

channelUP     CON 144
```

```

channelDOWN      CON 145
volumeUP         CON 146
volumeDOWN      CON 147
OK               CON 148      'Mute
Menu             CON 224
ENTER           CON 139
RETRN           CON 187
Power           CON 149
Temp            VAR BYTE
Temp2           VAR BYTE
WhichLED        VAR BYTE
FlashRate       VAR BYTE      'higher = slower
IRpulse_length VAR WORD(13)
xx              VAR BYTE
y               VAR BYTE
Command         VAR BYTE
LEDFLag         VAR BYTE(11)    'go to 11 since we need 0-10 (11
elements)
SEROUT Serial_out,n9600,[13,10,13,10,13,10,"d. bodnar ver 1.9 IR Sensor to
control",13,10]
SEROUT Serial_out,n9600,[13,10,"Black Board Blinker- 02-01-12",13,10]

DATA @0,2,200          'start with LED reacting to #1 on remote
READ 0, WhichLED      'remember which it is programmed to be
READ 1, FlashRate     'not currently being used
FOR Temp=0 TO 10:LEDFLag(Temp)=0:NEXT Temp ' clear array
Top:
IF LEDFLag(WhichLED)=2 THEN
    TOGGLE LED:TOGGLE LED2
ENDIF
Command=0
GOSUB GetIR
IF Command=Power THEN GoToSleep:
IF Command = RETRN THEN ' show unit #
    LOW LED:LOW LED2:PAUSE 400
    SEROUT Serial_out,n9600,["Blinking # ",#WhichLED,13,10]
    FOR Temp=1 TO WhichLED
        HIGH LED:HIGH LED2:PAUSE 200:LOW LED:LOW LED2:PAUSE 200
    NEXT Temp
    PAUSE 500
    IF LEDFLag(WhichLED)=1 THEN HIGH LED:HIGH LED2
ENDIF
IF Command = Menu AND LEDFLag(WhichLED)=2 THEN 'LED must be flashing to
change #
    SEROUT Serial_out,n9600,["@ Change #",13,10]
    LOW LED:LOW LED2:PAUSE 400
    FOR Temp=1 TO WhichLED
        HIGH LED:HIGH LED2:PAUSE 200:LOW LED:PAUSE 200
    NEXT Temp
    PAUSE 500
    StayHereTillRightNumberHit:
    GOSUB GetIR
    Temp2=Command-127
    SEROUT Serial_out,n9600,[ "new number is ",#Temp2,10,13 ]
    IF Temp2<11 AND Temp2>0 THEN
        WhichLED=Temp2

```

```

SEROUT Serial_out,n9600,[ "WhichLED= ",#WhichLED,10,13 ]
WRITE 0, WhichLED
FOR Temp=1 TO WhichLED
    HIGH LED:HIGH LED2:PAUSE 200:LOW LED:LOW LED2:PAUSE 200
NEXT Temp
LEDFLag(WhichLED)=0      'start with led off
PAUSE 200
GOTO Top:
ELSE
GOTO StayHereTillRightNumberHit
ENDIF
ENDIF
Temp2=Command-127      'convert to which number
SEROUT Serial_out,n9600,[ "T2,wled, arry ",#Temp2," ",#WhichLED,"
",#LEDFLag(Temp2)," ",#LEDFLag(WhichLED),10,13]
IF Temp2 = WhichLED THEN
    LEDFLag(Temp2)=LEDFLag(Temp2)+1
    IF LEDFLag(Temp2)=3 THEN
        LEDFLag(Temp2)=0
        LOW LED:LOW LED2
        PAUSE 200
    ELSE
        HIGH LED:HIGH LED2
        PAUSE 200
    ENDIF
    SEROUT Serial_out,n9600,[10,13, "Got It ledflag()
",#LEDFLag(Temp2),10,13,10,13 ]
ENDIF
GOTO Top

GoToSleep:
SEROUT Serial_out,n9600,[10,13,"@Sleep",10,13]
FOR y= 255 TO 0 STEP -3      'dim showing going to sleep
    PWM LED, y,3
    PWM LED2,y,3
NEXT y
SkipDim:
LOW LED:LOW LED2
LongSleep:
SEROUT Serial_out, n9600, ["Z", 10,13]
PAUSE 1000
INTCON.0 = 0      'Reset the Port change bit
gpio.5=0          'reset switch pin
IOC = %00100000 'enable INTERRUPT ON CHANGE on gpio 5
SLEEP 65000      'in seconds
IF intcon.0=0 THEN GOTO LongSleep
INTCON.0 = 0      'Reset the Port change bit
IOC=0
PAUSE 100
redo:
Command = 0:GOSUB GetIR
IF Command=0 THEN redo
SEROUT Serial_out, n9600, ["key = ",#Command, 10,13]
IF Command <> Power THEN
    SEROUT Serial_out, n9600, ["Not Power", 10,13]
    PAUSE 200

```

```

    GOTO SkipDim ' stay asleep unless Power hit again
ENDIF
FOR y= 0 TO 255 STEP 3 'brighten LED to show waking up
    PWM LED, y,3
    PWM LED2,y,3
NEXT y
IF LEDFLag(WhichLED)=1 THEN HIGH LED:HIGH LED2
SEROUT Serial_out, n9600, ["xZ", 10,13]
INTCON.0 = 0 'Reset the Port change bit
PAUSE 1000
GOTO Top:

'NOTE - numbers seem to vary with IR receivers - this set (240 / 120) seems
' OK with all three receivers that I have
GetIR:

PULSIN irin ,0, IRpulse_length(0)
IF IRpulse_length(0) < 240 THEN RETURN

FOR xx=1 TO 12
    PULSIN irin,0,IRpulse_length(xx)
NEXT xx

displaybits:
IF IRpulse_length(1) < 120 THEN
    Command.BIT0 = 0
ELSE
    Command.BIT0 = 1
ENDIF
IF IRpulse_length(2) < 120 THEN
    Command.BIT1 = 0
ELSE
    Command.BIT1 = 1
ENDIF
IF IRpulse_length(3) < 120 THEN
    Command.BIT2 = 0
ELSE
    Command.BIT2 = 1
ENDIF
IF IRpulse_length(4) < 120 THEN
    Command.BIT3 = 0
ELSE
    Command.BIT3 = 1
ENDIF
IF IRpulse_length(5) < 120 THEN
    Command.BIT4 = 0
ELSE
    Command.BIT4 = 1
ENDIF
IF IRpulse_length(6) < 120 THEN
    Command.BIT5 = 0
ELSE
    Command.BIT5 = 1
ENDIF
IF IRpulse_length(7) < 120 THEN
    Command.BIT6 = 0

```



```

ELSE
  Command.BIT6 = 1
ENDIF
IF IRpulse_length(8) < 120 THEN
  Command.BIT7 = 0
  ELSE
  Command.BIT7 = 1
ENDIF
IF Command.BIT7 = 0 THEN 'Bit 7 is one of the device bits
  Command = Command + 1
ENDIF
IF Command = 10 THEN
  Command = 0
ENDIF
RETURN

```

Power Management

Since this device is operating from a very small power supply power management is important if you want to avoid replacing batteries very often. This can be done in several ways. The first is to use the LED in flash mode as often as possible. As the table below shows when the LED is full on the device draws about 10 ma at 3 volts. When it flashes that power demand drops to half of that amount.

The second thing that can be done is to put the device into "sleep" mode when it is not in operation. With the LED off but with the unit still operating it draws 0.8 ma. This is a very small amount of current, but that can be cut in half by putting the microcontroller into sleep more.

LED Status	Current @ 3 v.
LED off	0.8 ma
LED on full bright	10 ma
LED blinking	about 5 ma
Sleep Mode	0.4 ma

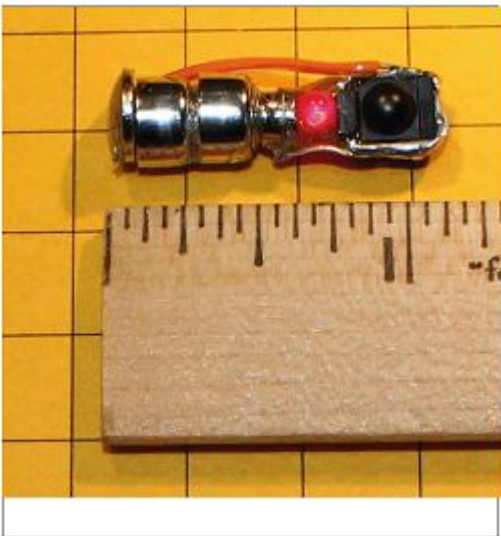
If a battery's mAh rating is known is it easy to estimate how long the unit will operate with it. Just divide the mAh rating by the current in whatever mode you are using. The time of operation, in hours, is the result. For example, if you use a 50 mAh battery and operate the unit with the LED full on the computation is $50 / 10 = 5$ hours. Increasing the size of the battery or decreasing the current consumption will increase the run time.

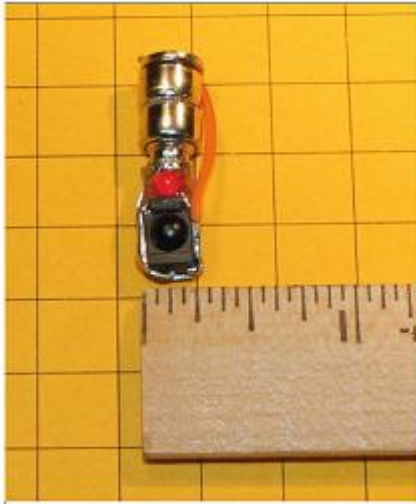
This photos show the completed unit. It is not housed in a capsule, per se, but could easily be so enclosed. The rare earth magnets that hold the batteries to the unit do a nice job of keeping the unit in one piece. The LED, microcontroller and IR sensor have been glued together after soldering with 5 minute epoxy.

From left to right the parts that make up the complete device are

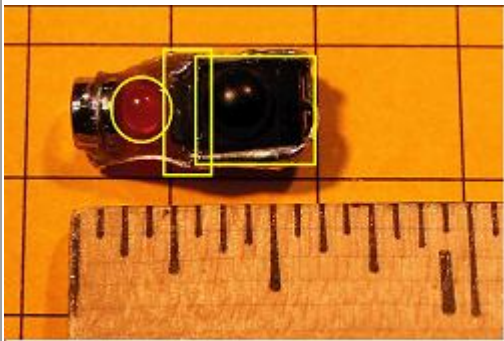
1. rare earth magnet with orange wire attached
2. two 1.5 volt cells with a small rare earth magnet between them
3. a larger magnet that is glued to the items to the right. It is also connected to the negative terminal on the microcontroller.
4. 3mm red LED
5. 12F683 microcontroller - it is only 2mm thick and can barely be seen to the right of the LED
6. Infrared sensor. Its dome is the part that must "see" the IR pulses.

You will note that the LED and IR sensor both face to the side so that the unit can be affixed to the black board on its side. If a vertical mounting is preferred the LED could be placed at the top facing up.

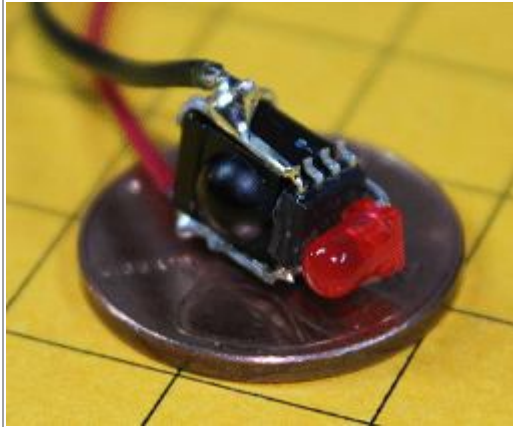




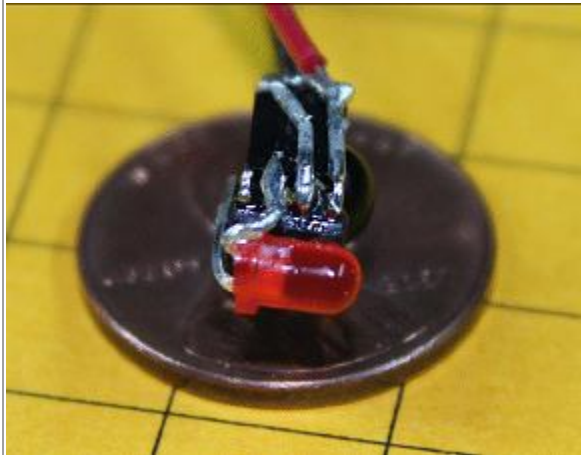
Here the LED is circled. The microcontroller has a rectangular box around it and the IR sensor is shown with a larger rectangle around it.



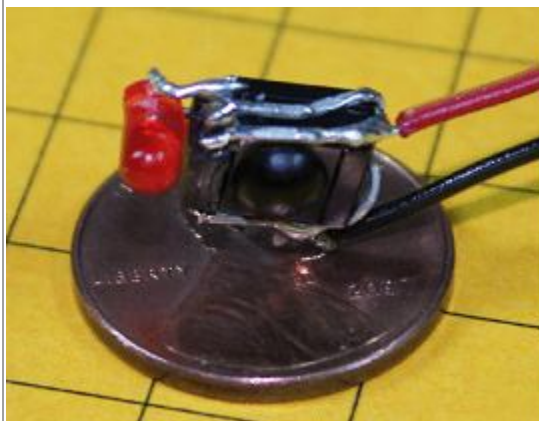
This image shows the three components wired together and sitting on a penny.

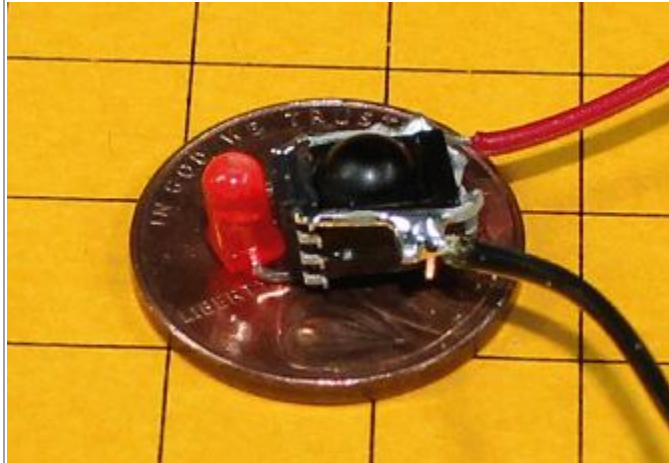


This view shows the same devices from the other side.



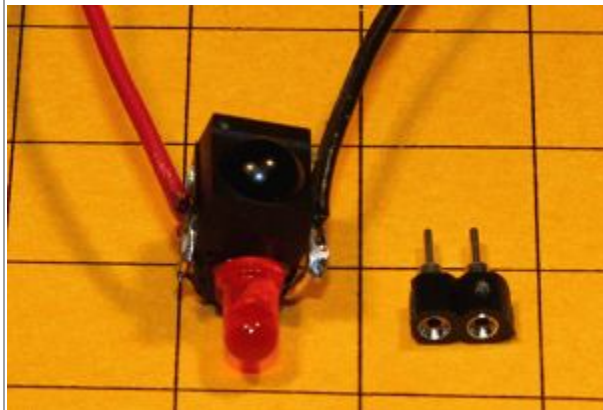
Note that the positive wire goes to 12F683 pin 1 and the negative goes to pin 8.





Vertical Design

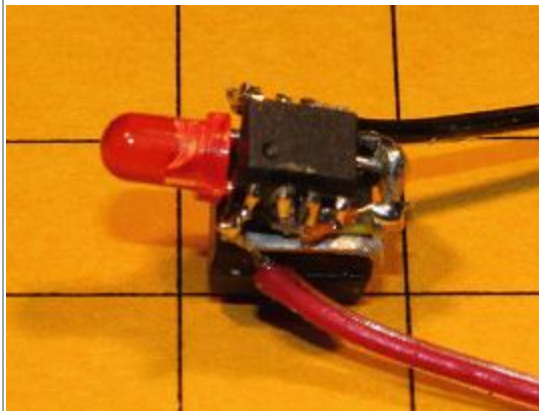
These photos show a different way that the components can be arranged to put the LED at the top of a vertically oriented unit. The batteries will be attached to the end opposite the LED. The small set of machined pins to the right are what the LED is inserted in. This allows you to easily change LEDs as the leads are not soldered. The machined pin socket is sandwiched between the IR sensor and the 12F683 and does not add any length to the unit.



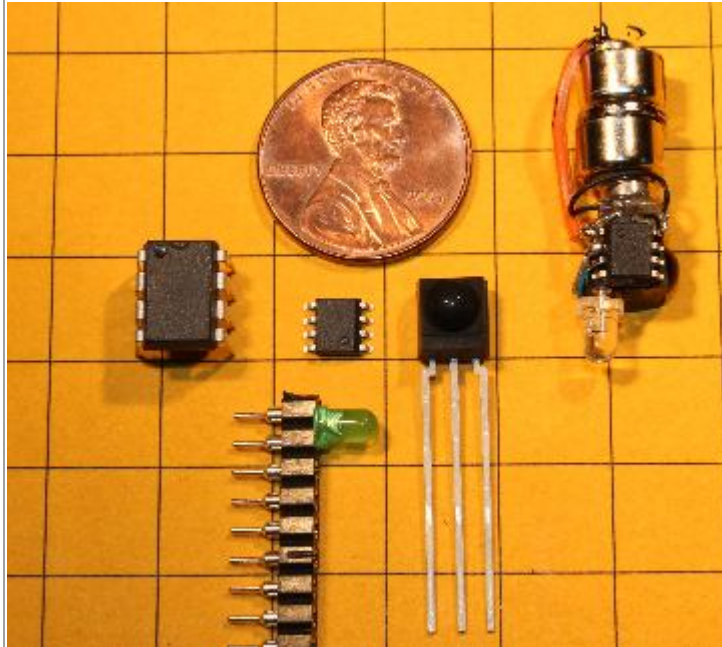
This close-up shows the socket that is used to hold the 3mm LED. LEDs do need to be inserted with the right polarity. Fortunately they are not harmed if inserted backwards so, if it doesn't work when put in one way just try the other!



This view clearly shows the 12F683 microcontroller with the LED socket sandwiched between it and the IR sensor. The magnet for the negative terminal of the battery will be glued to the right end.



This photo shows a non-surface mount 12F683 next to the surface mount version. You can also see the IR sensor and a string of the socket connectors that hold the LED's leads.



I was able to find empty gelatin capsules on [eBay](#). The circuit is shown inside of one of the capsules. To its right is an empty capsule top and bottom. The ones I used are from eBay. They are size "000" and are made from gelatin. This means that they are water soluble so they must be kept dry or you need to coat them with clear paint or varnish. They are about 9mm in diameter and about 26mm long. When inserting into the capsule make sure the two cells are aligned evenly with the magnets.



Note that there is a magnet between the two cells (batteries) - that magnet must be aligned so that it will stick to the other magnets. If it is put in upside down the magnets will not adhere very well. It is also VERY important that the positive ends of the cells point to the red wire and that the smaller terminal (negative) goes towards the circuitry. This can be seen clearly in this photo.



Battery graph - 10 second readings (2200 points = a bit over 6 hours) - worked throughout this test - LED solid on except for peaks where remote was used to change mode to see if it was still active

still working @ 2 volts but batteries failing fast at that point. IR remote stopped working at 1.94 volts - quite a bit below its rated minimum.



After completing a number of the surface mount versions of this circuit it occurred to me that some folks don't need the device to fit into a small capsule and might want to try working with somewhat larger components.

