433MHz RF Transmission and Reception system

Designed by RM

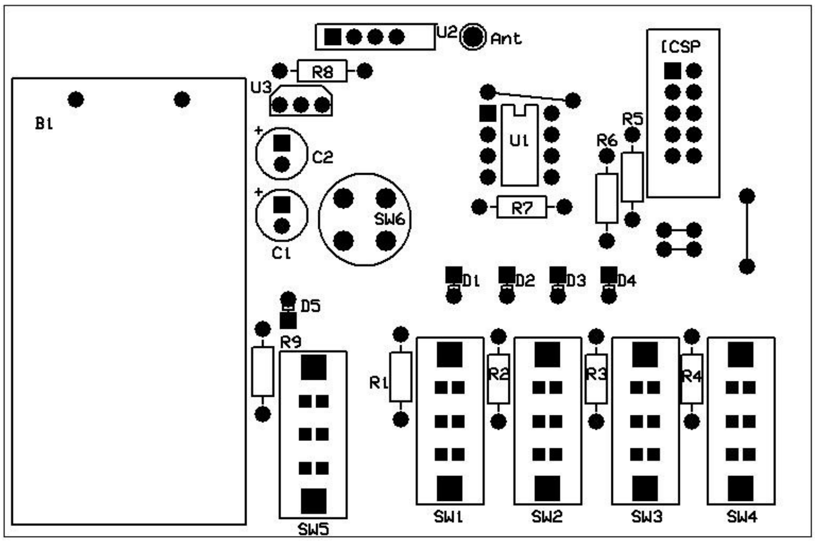
This project file details a microcontroller based data transmission system utilizing rudimentary 433MHz transmission modules. The system consists of a transmitter and receiver, both using ATTiny13 8-pin microcontrollers. The method used for transmission is a Pulse-width based method, and the 4-bit data words that are transmitted are generated using switches and displayed on LEDs.

It should be noted that this system could be adapted, with minor additions or modifications, to perform more useful tasks such as turning on remote equipment.

Contents

Transmitter

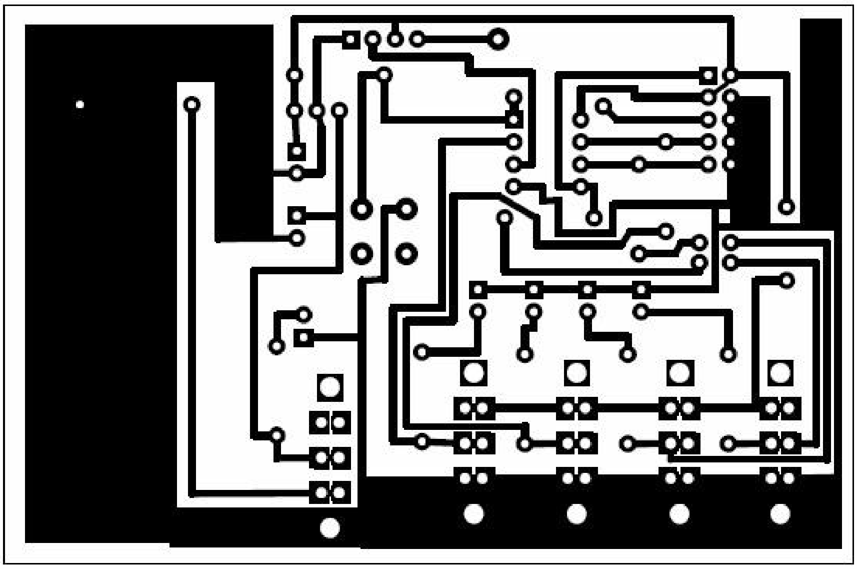
Basic Digital Transmitter



Parts:

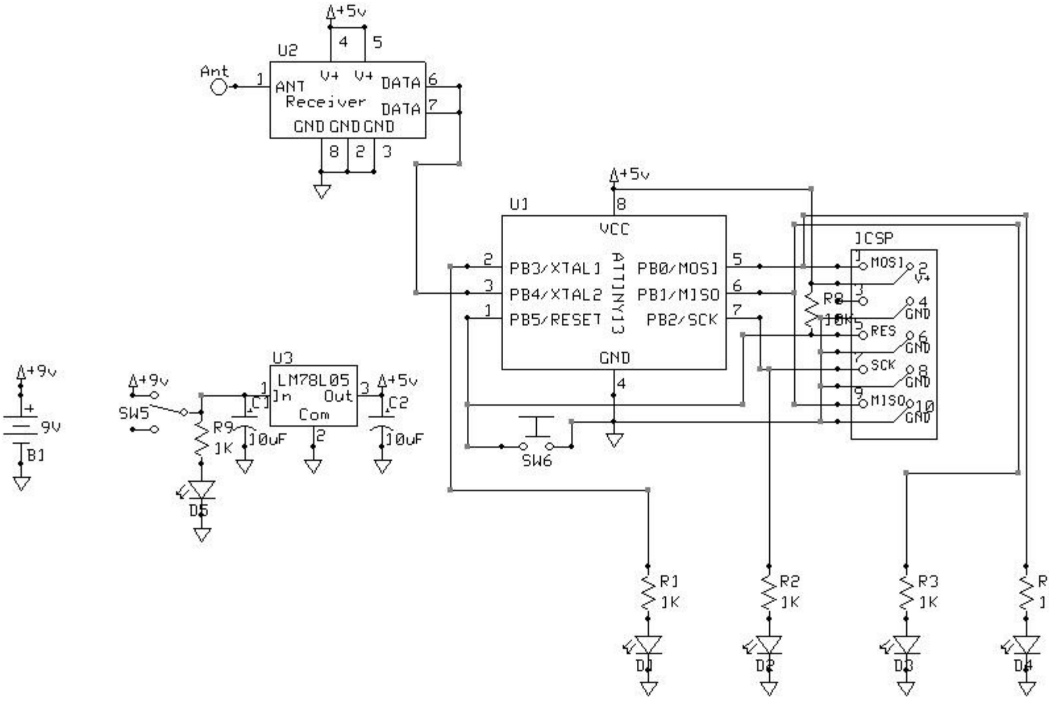
R1	1K
R2	1K
R3	1K
R4	1K
R5	10K
R6	10K
R7	10K
R8	10K
R9	1K
C1	10uF
C2	10uF
D1	LED (Bit 3)
D2	LED (Bit 2)
D3	LED (Bit 1)
D4	LED (Bit 0)
D5	LED (Power)
U1	ATTINY13
U2	433MHz RF Transmitter Module
U3	LM78L05
Ant	Simple wire antenna
B1	9V PCB battery holder
ICSP	10pin Header
SW1	DPDT PCB Submini Slide switch
SW2	""
SW3	""
SW4	""
SW5	""
SW6	PCB Pushbutton

The specified 433MHz antenna length is 17.2cm.

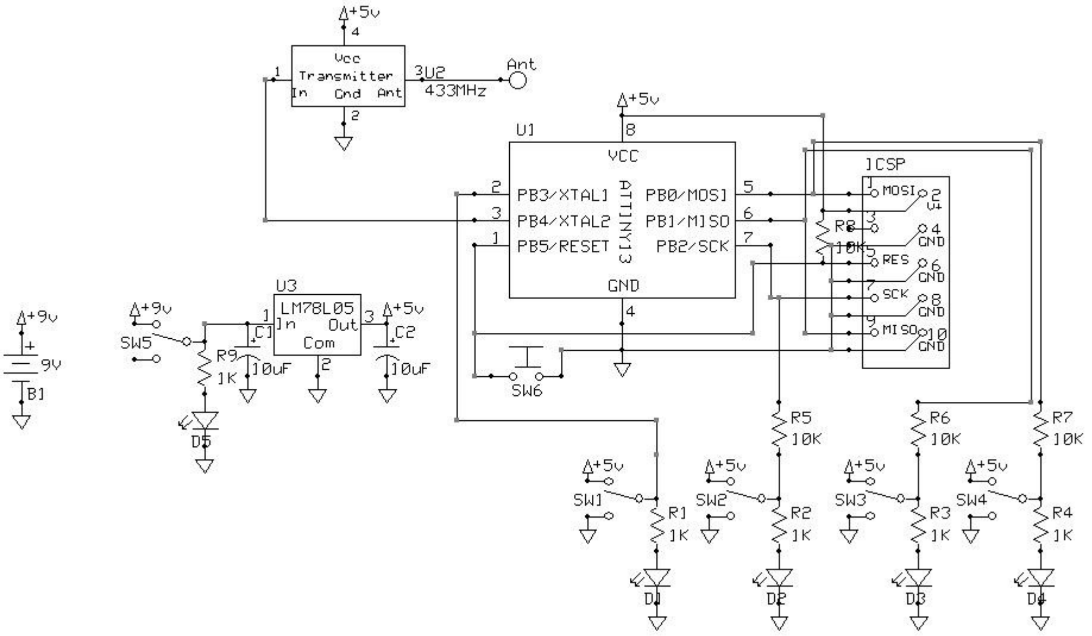


Note: PCB not to scale. When scaling for printing, width from border to border should measure 3.8 Inches, and height from border to border should measure 2.5 Inches.

Receiver

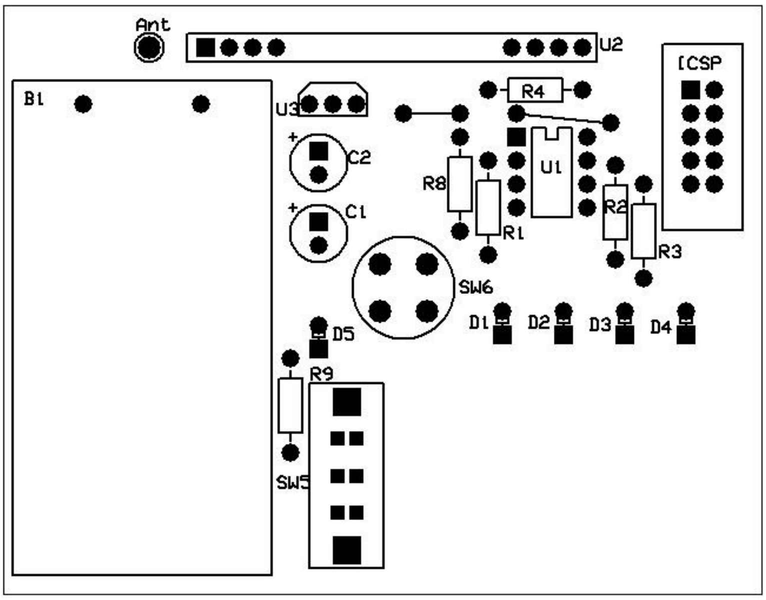


Transmitter



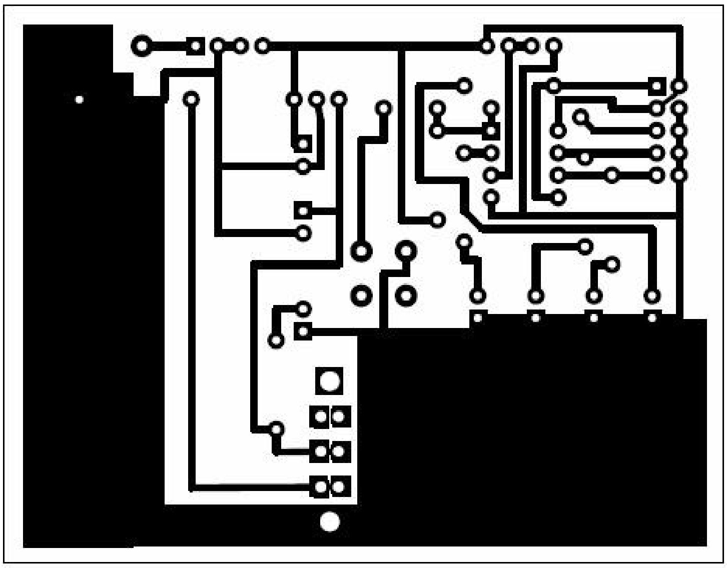
Receiver

Basic Digital Receiver



Parts:

R1	1K
R2	1K
R3	1K
R4	1K
R8	10K
R9	1K
C1	10uF
C2	10uF
D1	LED (Bit 3)
D2	LED (Bit 2)
D3	LED (Bit 1)
D4	LED (Bit 0)
D5	LED (Power)
U1	ATTINY13
U2	433MHz RF Receiver module
U3	LM78L05
Ant	Simple wire antenna
B1	9V
ICSP	10pin Header
SW5	DPDT PCB Submini Slide switch (Power)
SW6	PCB Pushbutton (Reset)



Note: PCB not to scale. When scaling for printing, width from border to border should measure 3.2 Inches, and height from border to border should measure 2.5 Inches.

Assembly Code

The programs for the receiver and transmitter were written using assembly language, and programmed into the microcontrollers using AVR Studio 4.12 and PongProg2000 (with ATTiny13 support). The programming hardware was a DIY parallel port programming dongle.

Transmitter Code:

```
.include "tn13def.inc"
.def temp = r16
.def temp2 = r17
.def count = r18
.def temp3 = r19
.def temp4 = r20

reset:
ldi     temp, 0b00010000
out     DDRB, temp
ldi     temp,low(RAMEND)
out     SPL,temp
ldi     temp, 0b00000101
out     TCCR0B, temp

main:
in      temp, PINB
ldi     count, 5

sbi     PORTB, 4 ;start pulse
rcall   delay
rcall   delay
rcall   delay
cbi     PORTB, 4
rcall   delay
rcall   delay
rcall   delay

sendloop:
dec     count
breq    endtrans
sbrs    temp, 3
rjmp    send0
send1:
sbi     PORTB, 4
rcall   delay
rcall   delay
cbi     PORTB, 4
rcall   delay
rcall   delay
lsl     temp
rjmp    sendloop
send0:
sbi     PORTB, 4
rcall   delay
cbi     PORTB, 4
rcall   delay
lsl     temp
rjmp    sendloop

endtrans:
rjmp    endtrans

delay:
ldi     temp3, 221
ldi     temp4, 16
delaya:
dec     temp3
brne    delaya
ldi     temp3, 221
dec     temp4
```

```
brne        delaya
ret
```

Receiver Code:

```
.include "tn13def.inc"
.def temp = r16
.def temp2 = r17
.def count = r18
.def data = r19

.equ startbitlength = 30
.equ bit1length = 21
.equ bit0length = 11
.equ startcutoff = 36
.equ bit1cutoff = 27
.equ bit0cutoff = 15
.equ pulsecutoff = 5

;PB0-3 - LEDs (out)
;PB4   - Data (in)

reset:
ldi         temp, 0b00001111
out         DDRB, temp
ldi         temp, RAMEND
out         spl, temp
ldi         temp, 0b00000101
out         TCCR0B, temp

main:
out         PORTB, data ;output data

sbis        PINB, 4     ;wait for start bit
rjmp        main

ldi         temp, 0
out         TCNT0, temp
parta:
sbic        PINB, 4
rjmp        parta

in          temp, TCNT0
cpi         temp, startcutoff ;test pulse size
brlo        partb
rjmp        main
partb:
cpi         temp, bit1cutoff
brge        partc
rjmp        main

partc:      ;decoding subroutine
ldi         count, 4
clr         data

partd:
sbis        PINB, 4     ;wait for bit
rjmp        partd

ldi         temp, 0
out         TCNT0, temp

parte:
sbic        PINB, 4
rjmp        parte

in          temp, TCNT0
cpi         temp, bit1cutoff
brlo        partf
rjmp        main

partf:
```

```
cpi         temp, bit0cutoff
brlo        partg
lsl         data
sbr         data, 1 ;put a 1 in data register
dec         count
breq        main        ;all data received
rjmp        partd       ;or continue decoding

partg:
lsl         data
cbr         data, 1
dec         count
breq        main        ;all data received
rjmp        partd       ;or continue decoding
```

How it works

Instructions

To operate the devices, you enter a 4-bit word using the switches on the transmitter, and then press the reset button. Providing you are within range, the 4-bit word will appear on the LEDs on the receiver.

The system works using a Pulse-width based method, which uses different sized pulses to indicate ‘start’ ‘1’ and ‘0’. The start bit indicates a new transmission word is arriving. The program cycles through each bit of the word, and creates the appropriate sized pulse for each one. The receiver times the lengths of each pulse, and then places the appropriate bit in the data register. The reason this method was used, was because the rudimentary RF modules used require the amount of ‘on’ and ‘off’ time to be near the same to function correctly.

Miscellaneous

Programming
If you do not have a commercial AVR programmer, you can build your own very simple one, using the details from the following website: http://www.tothemax.web1000.com – under "PIC and AVR". You will need to download PonyProg2000 and get the ATTiny13 support executable.

Parts
The 433MHz RF Modules are available from Jaycar Electronics (http://www.jaycar.com.au), and cost $AUS20 a pair.

Hex Files
Compiled Hex files for use with PonyProg are available at these addresses:
http://www.geocities.com/race_driver205/trans.hex
http://www.geocities.com/race_driver205/receiv.hex

Use of this document
If you want to use the information in this document in commercial designs, go for your life. If you make millions and retire in a beach-front villa with a super-model wife and a Ferrari, I'm cool with that.

Modifications
The most obvious modification is to change the "rjmp endtrans" instruction in the transmitter program to "rjmp main". The will mean you don't have to press the pushbutton to send each data word, and flicking the switches will change the receiver outputs instantly.

Other Uses
The transmission method could easily be modified to allow for transmission of 8-bit or greater words. The key things to change would be the 'count' register initial value, and the bit position that is read from.

Help
If you have any questions, please forward them to tothemax6@hotmail.com. I guarantee that I built these devices myself and that they work correctly, so debugging questions are regrettably unlikely to be answered.

The Finished Gear