

# Atmel AVR 2004 Design Contest Entry A3720

## IR Train Controller

### Abstract

In the Christmas sales last year my wife picked up a battery operated train set very cheaply. My son thought it was great, but it was evident that this toy would not last long in the hands of a 3-year-old! It was soon designated as "Daddy's train" in an attempt to control its use, but Daddy enjoyed playing with it too!

The main potential for damage was while starting and stopping the train since it had to be held still while operating the switch on the side of the engine. Often with the train held still, the engine would move the tracks instead! The solution was to implement some form of remote control, and thereby remove the need to touch the train at all.

The train was of cheap plastic. It had a couple of carriages, and ran on plastic tracks. The batteries were contained in the train engine itself. A forward-off-reverse switch on the side of the engine operated it. Appropriate whistles, track-noises and bell sounds were generated from an internal sound generator circuit and speaker, and the headlamp worked using a small grain-of-sand light bulb.

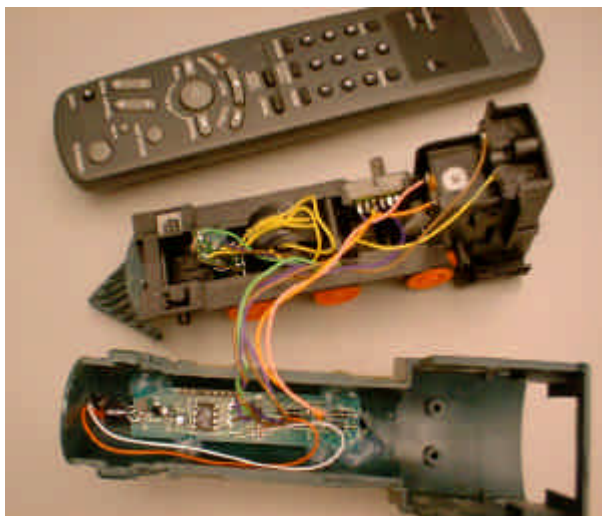
An IR remote from a long dead VCR was available, so what was needed was a decoder and the appropriate electronics to control the train motor, sound and light. An Atmel ATtiny15L was used to decode the output from an integrated IR remote receiver (this detected the 38kHz IR signal from the remote control), and to then control the motor (in either direction using an H-bridge), the sound generation circuit, and the light. The whole circuit needed to operate on 3V, which required a voltage doubler and regulator for the IR receiver device, which needed 5V.

The software was developed to decode the IR remote signals and control the train appropriately. The remote control used conformed to the NEC/APEX/HITACHI/PIONEER IR coding specification. Through simple timing routines it was straightforward to develop a decoding application for it. The main application loop handled this, with some background state machines to control the train motion and sound output.

The circuit was assembled on to a specially design double sided PCB using mostly SMD components to keep the size down. Once assembled and the software developed, the PCB was mounted into the 'boiler' of the train engine with the IR receiver visible through a small hole on top. The battery holder, switch, motor, sound generator, and light of the train were wired to the IR decoder PCB.

The end result worked well. The train modifications are not visible. The train now moves, whistles, and lights up on command. Daddy and son are very happy with the result!

### Project Photographs

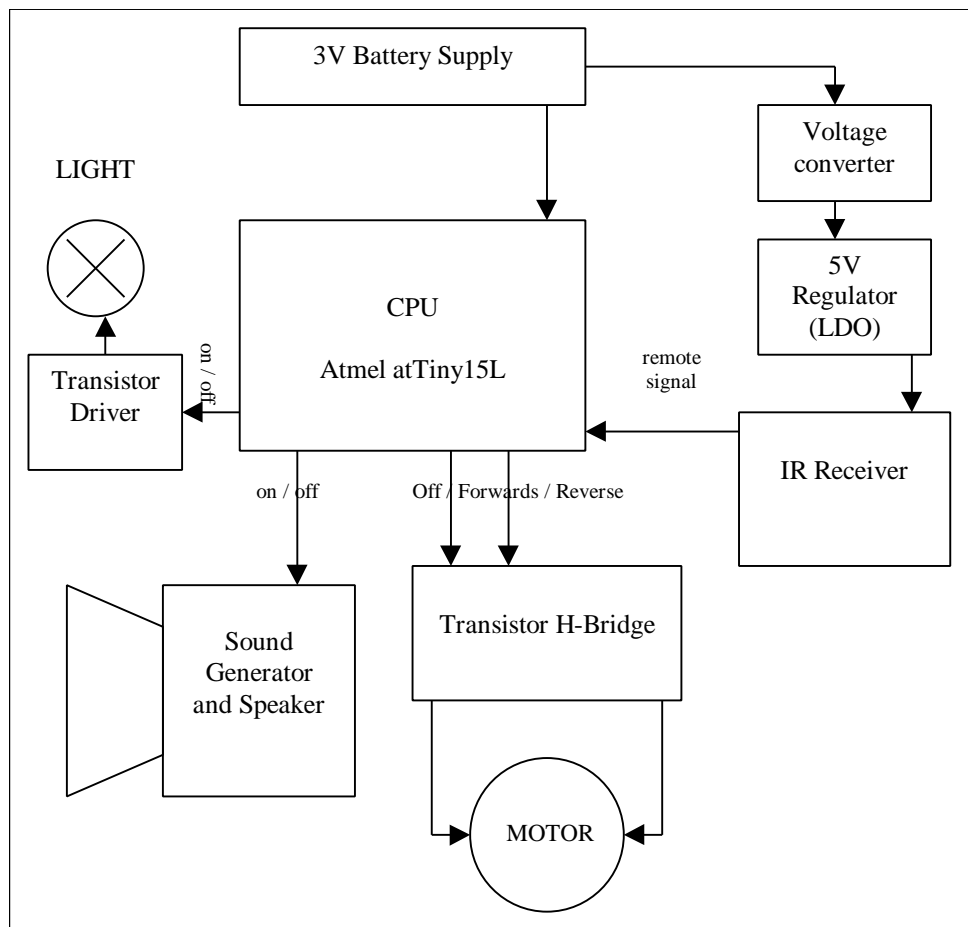


The PCB wired into the train



The final re-assembled train

## Block Diagram



Block Diagram of the Remote Control Train hardware

## Software Code Sample

```

;*****
;* Motor State-Machine
;*
;* The motor is controlled by an H-bridge.
;* This state machine controls the H-bridge signals based on the desired
;* motor mode (stopped, forwards or backwards).
;* The state machine ensured there is a small STOPPED delay when swapping
;* between forwards and backwards.
;*
;* The state machine controls the FWD and REV outputs
;* The input is the MOT_MODE register (motor mode) = 0 (stopped),
;* 1 (forward) or 2 (backwards)
;*
;*****

MotorSM:
    ; switch on state
    cpi    MOT_STATE,0
    breq  MT_State0
    cpi    MOT_STATE,1
    breq  MT_State1
    cpi    MOT_STATE,2
    breq  MT_State2
    cpi    MOT_STATE,3
    breq  MT_State3
    rjmp  MT_ToState0

; State0 - Motor Off
MT_ToState0:
    ldi    MOT_STATE,0
    ; turn off motor
    STOPPED
  
```

```

MT_State0:
    ; if forward request, goto state 1
    cpi    MOT_MODE,1
    breq   MT_ToState1
    ; if reverse request, goto state 2
    cpi    MOT_MODE,2
    breq   MT_ToState2
    rjmp   MT_End

; State1 - Forward
MT_ToState1:
    ldi    MOT_STATE,1
    ; go forwards
    FORWARDS
MT_State1:
    ; if stop request, goto state 3
    cpi    MOT_MODE,0
    breq   MT_ToState3
    ; if reverse request, goto state 3
    cpi    MOT_MODE,2
    breq   MT_ToState3
    rjmp   MT_End

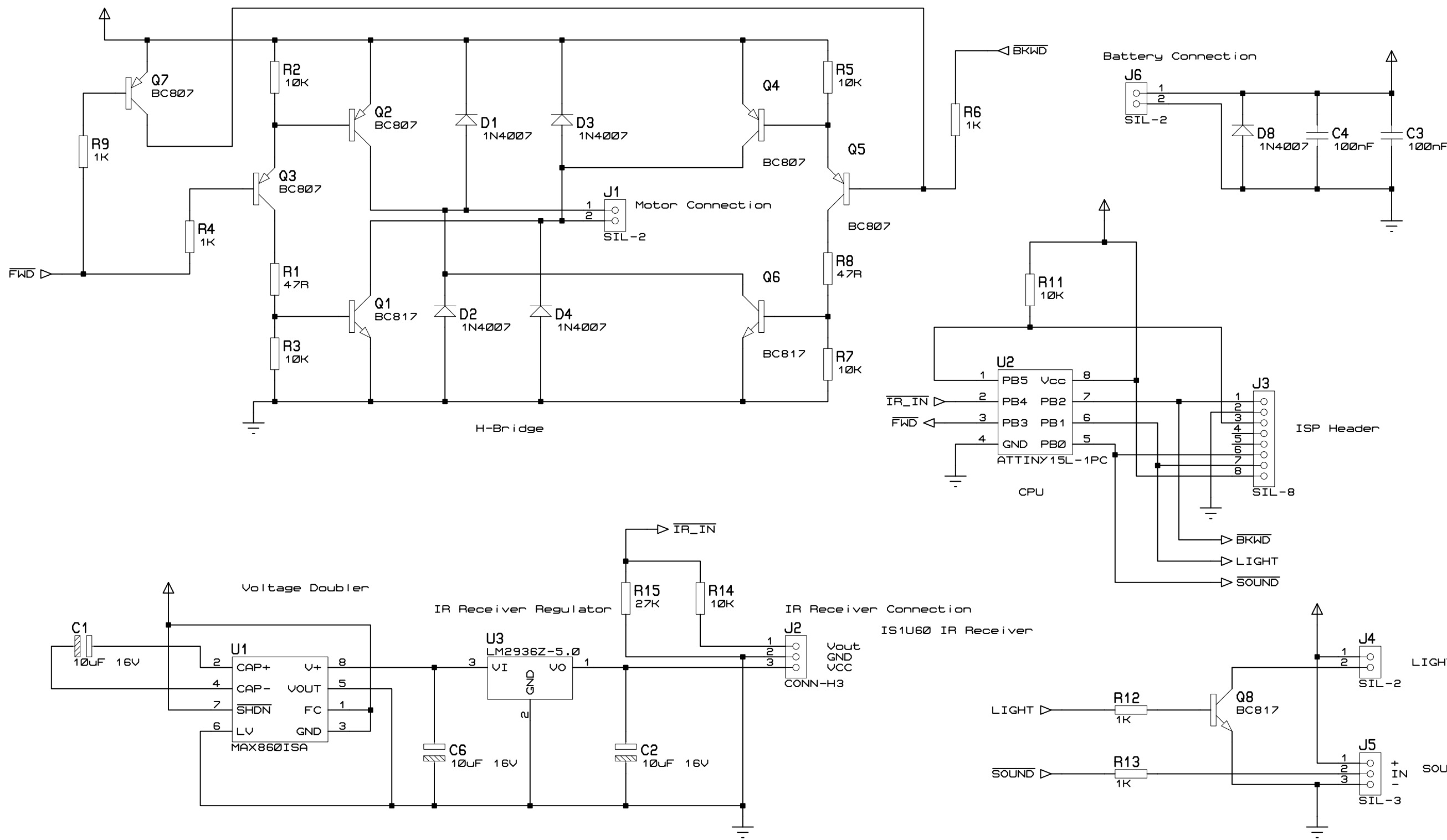
; State2 - Backwards
MT_ToState2:
    ldi    MOT_STATE,2
    ; go backwards
    BACKWARDS
MT_State2:
    ; if stop request, goto state 3
    cpi    MOT_MODE,0
    breq   MT_ToState3
    ; if forwards request, goto state 3
    cpi    MOT_MODE,1
    breq   MT_ToState3
MT_End2:
    rjmp   MT_End

; State3 - Pause
; S/M always goes through this state when motor is stopped or reversed.
MT_ToState3:
    ldi    MOT_STATE,3
    ; stop motor
    STOPPED
    ; start timer
    ldi    TIM20mSB,10 ; 200mS seconds
MT_State3:
    ; if not timeout, exit
    tst    TIM20mSB
    brne   MT_End2
    ; if forward request, goto state 1
    cpi    MOT_MODE,1
    breq   MT_ToState1
    ; if reverse request, goto state 2
    cpi    MOT_MODE,2
    breq   MT_ToState2
    ; else goto state 0
    rjmp   MT_ToState0

```

## Circuit Diagram

- refer to separate document - TrainIR\_Schematic.pdf



TITLE:	Train Remote Receiver	DATE:	29/06/04
BY:		PAGE:	1/1
		REV:	1.1