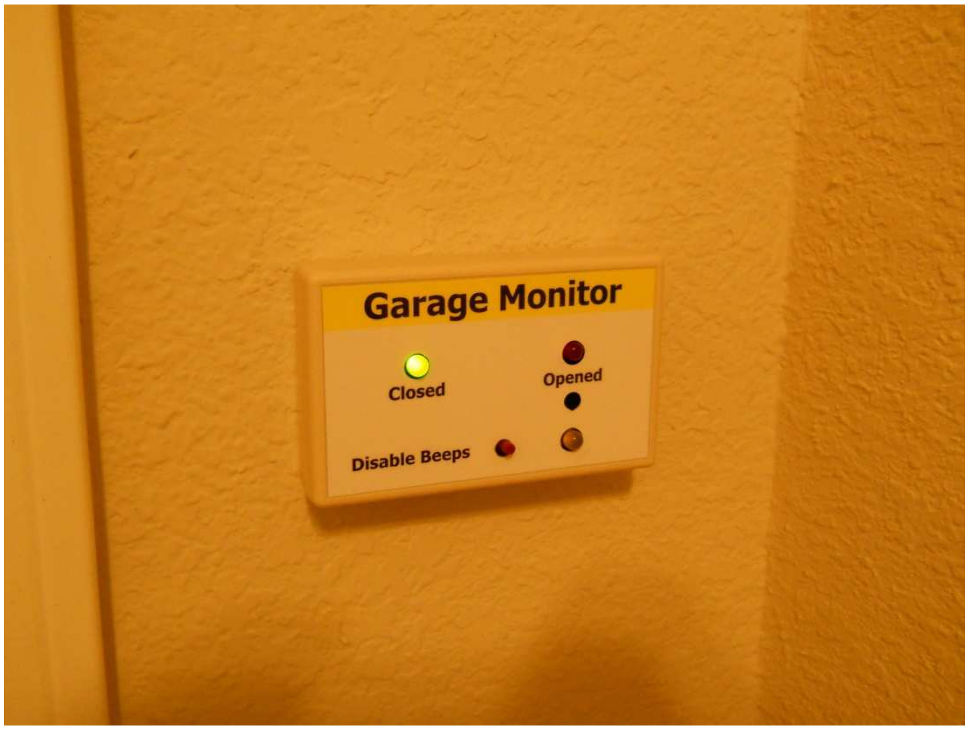
Garage Monitor



Disclaimer:

I make no guarantee(s) about this project and am not liable for any issues that may arise from the information provided here.

About:

This project is highly influenced by the many disappointments I experienced in a garage monitor that I ordered off the Internet before I knew how to use microcontrollers. It was wireless which should have been a good thing except that it wouldn't work even at minimal ranges such as 20 feet. It would often indicate that the door was open when it was not. If the unit lost power and restarted, it would start up with an alarm which at 3am might make you think someone is in the garage when they shouldn't be. It also used expensive replacement batteries for the transmitter which needed constant changing. Needless to say it was far more trouble than it was worth and found its way to the trash can at an expense of $40.

This project uses a wired garage door sensor, so no issues with transmission distance or radio interference. If the unit loses power and comes back on, it will do so quietly. I implemented a watchdog timer in the software to reset it in case it gets hung for any reason to ensure that the LED door status indicator is correct. It will repeatedly sound an alarm if the door is left open too long, and has an optional door chime that indicates when the door changes state. Instead of tempting the user to unplug it to disable it when necessary, it also has a disable beep feature that will disable all sound for one to eight hours. All this in a polished package that looks great on the wall and is easily to build!

Features:

Door Indicator - When the garage door is closed a green LED is on solid. When the garage door is opened a red LED is blinking. This feature will allow you to check the door state without going to the garage depending on where you put your garage monitor.

Door Alarm - If the garage door is left open longer than 5 minutes, it will sound three 1 second beeps. It will repeat every 3 minutes thereafter until the door is closed or the sound is disabled.

Door Chime - If enabled, this feature will sound two or three 100ms beeps when the garage door changes state. If the door is closed, it will sound two 100ms beeps, and if the door is opened it will sound three 100ms beeps. To toggle this feature (it is remembered in EEPROM), hold the beep disable button down while powering on. Both green and red LED will be on solid and the yellow LED will indicate if this feature is enabled or not. Tap the beep disable button to toggle it. Hold the beep disable button to exit or power cycle the unit to exit.

Beep Disable - This feature allows the user to disable all sound from the unit for one to eight hours. If you want to work in the garage and plan on leaving the door open for a couple of hours, you can use the beep disable to prevent any alarms from coming from the unit. Without this feature, someone likely would have unplugged the unit and then possibly forgotten they unplugged it. The nice thing about this feature is that within one to eight hours, the unit is back to normal.

This project could also be used for something instead of a garage door too; anything that has two states and you want an indicator of which state.

Installation Tasks:

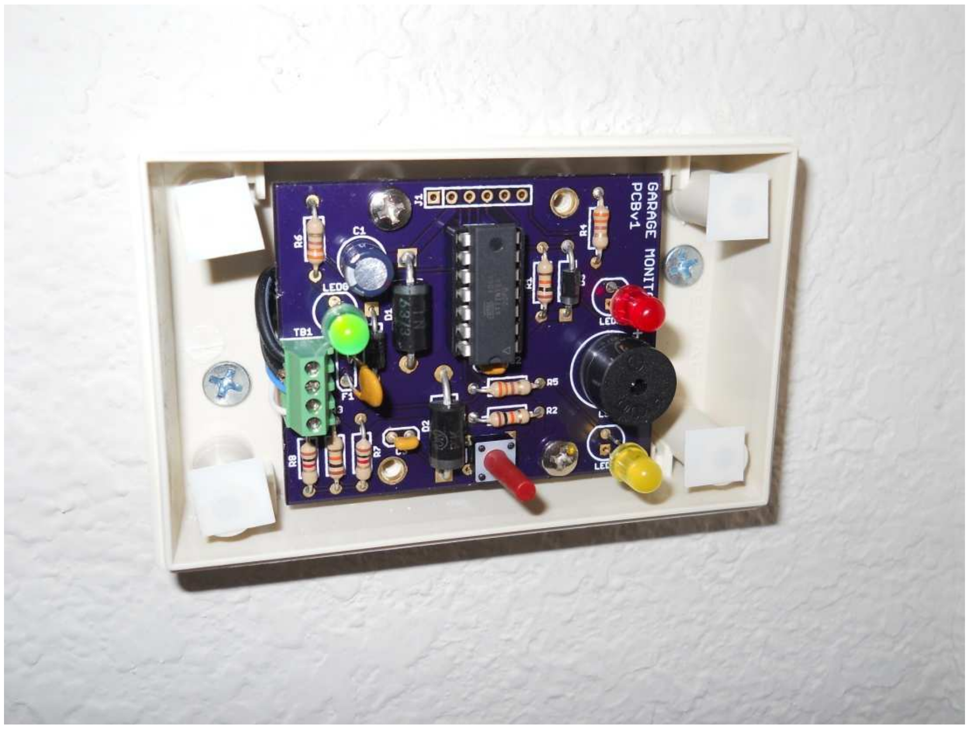
Since this project uses a wired sensor, you need to be able to run a wire (2 conductors) between the garage door and the garage monitor. I am using an alarm system style magnetic reed switch to sense the garage door being open or closed. Be sure to disconnect an automatic door opener if you have one and slowly move the door to test your sensor location. Stay away from the track as no matter where you think it might work, something will likely rub or crush the sensor. I know as I tried 5 different spots before giving up on a track mounted sensor. Also, test the sensor for dead zones. My sensor when lined up perfectly had a 1/16" vertical play area that did not activate the switch, but offsetting it by 5/16" and it worked perfectly every time. You can just connect it to a multimeter in ohms or continuity for testing.

You need to think about how you plan to power the unit. It does not have an onboard power supply so you need to feed it a regulated 5V DC. I have a recommended switching adapter in the parts list if you don't have something already.

I've included Gerber files in case you want to have a professional PCB made. The best place to get this done is the dorkbotpdx PCB service linked here:

http://dorkbotpdx.org/wiki/pcb_order

This is an awesome service that provides you 3 boards delivered in the USA for $5 per square inch. No setup fee and no shipping. The PCB is 3.96 square inches so for $19.80, you will receive three of these boards shipped to your door. They are excellent quality, both sides with solder mask and silkscreen. You may not need three, but I doubt you will find another PCB manufacturer that will build you even one non solder masked/non silkscreened PCB for this price. Build two extra units and sell them to your friends! You can of course skip the manufactured PCB route and build it using some generic board according to the schematic.



You need to decide if you are going to program your microcontroller on board or off board. The PCB includes a 1x6 inline ISP header for on board programming. If you use a socket on the board, it makes it easy to pull the microcontroller, plug it in somewhere else, and then reinstall it. If you are programming it off board and you plan to solder it in place (no socket) and you don't have a programming header, then you will need to program it ahead of time and **set its fuses properly**.

The face label is on the final page of this document and is made by printing it onto a white label. Make sure when you print this that Adobe has **page scaling on none** so it is not resized in any way. It should measure 2" x 3.35". I usually use Avery 5625 full sheet labels. I then cover the label with crystal clear tape. You will likely need some 3" wide for this label as I think it is just too tall for 2". I then use a cutting board with a roller cutter to cut it out just on the inside of the black lines. They are your guide on how to align the cutter. Practice on some regular paper with the label printed first. The final step is using a hole punch to cut out the LED holes and switch hole.

Use the holes in the face label to punch the face of the enclosure so it can be drilled. **DO NOT stick it to the face yet.** Drill holes for the LED's and switch. It may be a bit big, but I used a 5/16" bit for the LED's and a hole directly over the beeper (so it can be loud) and a 3/16" bit for the switch. Once the face is drilled, you might clean up the holes using a chamfer if you have one. Finally clean off the face and stick the label.

You'll note that there is no way to screw the face down with the enclosure on the wall and the picture above of how I mounted the enclosure to the wall does not give this option. I used thin double stick tape on all 4 posts to keep the face on. This works extremely well and allows you to mount the unit to the wall in a solid way. An alternative is to use the screws that came with the enclosure and then Velcro the enclosure to the wall.

<u>Software Notes:</u>

I built this with AVR Studio 4.18 SP2 with WinAVR 20100110, but it should work with most any modern version of AVR Studio. Source and compiled files are provided.

<u>It is critical</u> that the fuses are set properly on the ATTINY24A. The CKDIV8 fuse must be unchecked. If you notice that the LED's are blinking terrible slow it is likely you forget to uncheck this fuse. Clearing this fuse takes the uC from 1 MHz to 8 MHz. You can optionally set BODLEVEL for brownout detection as well. I've tested my PCB at brown-out detection at 4.3V and it seems to work just fine.

This results in these fuse byte values: Extended=0xff, High=0xDC, Low=0xE2

I have tried to comment the source as well as I could, so reviewing the garagealarm.c file should be fairly self explanatory.

I tried to put most constants such as the alarm time (5 minutes) and repeat alarm time (3 minutes) as well as many others in the first area of the file using defines. This should allow it to be easily tweaked to your needs.

I also commented on the volatile variables (they also have a preceding v_) as to where and how they are used. There is one timer ISR that is executed every millisecond. It keeps time, handles input debouncing, led status (blinking), and controls the beeper. The reason I put the led status and beeper in the ISR is because these events can continue to time properly no matter what is going on in the main loop.

There is a function beep that sets up a beep sequence. You specify the number of milliseconds and how many beeps and it will "communicate" with the ISR using volatile shared variables to get the request handled.

The main loop disables the watch dog timer (if enabled at startup), enables pull-ups on input pins, enables outputs, enables the timer ISR and global interrupts, reads the door chime setting from EEPROM, handles changing the door chime setting in EEPROM, and then handles everything else. I am using the watch dog to reset the uC if the main loop or something else is halted. I implemented the watchdog because nothing would annoy me more than seeing the status on the indicator and it NOT being correct.

<u>Hardware Notes:</u>

I designed it for a 5V input and included a PTC fuse, reverse polarity diode, main capacitor, and TVS diode for ESD protection. You will want to use a power source that provides a regulated 5V. The PTC is a 100mA, but I doubt it will ever use more than 50mA at one time.

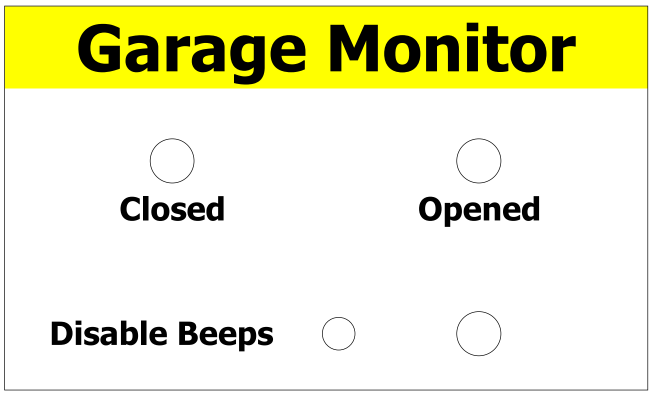
I put this project on a 14 pin TINY24A, but I almost put it on an 8 pin uC. I would have had to combine the input button and a LED using a pin sharing technique, but decided to just keep it simple and use a larger uC. You could likely adapt it to an 8 pin uC if you wanted to without too much difficulty. The uC has a couple of supporting components: bypass capacitor, reset pull-up resistor, and reset protection diode. My PCB has an ISP port for programming the uC on the PCB.

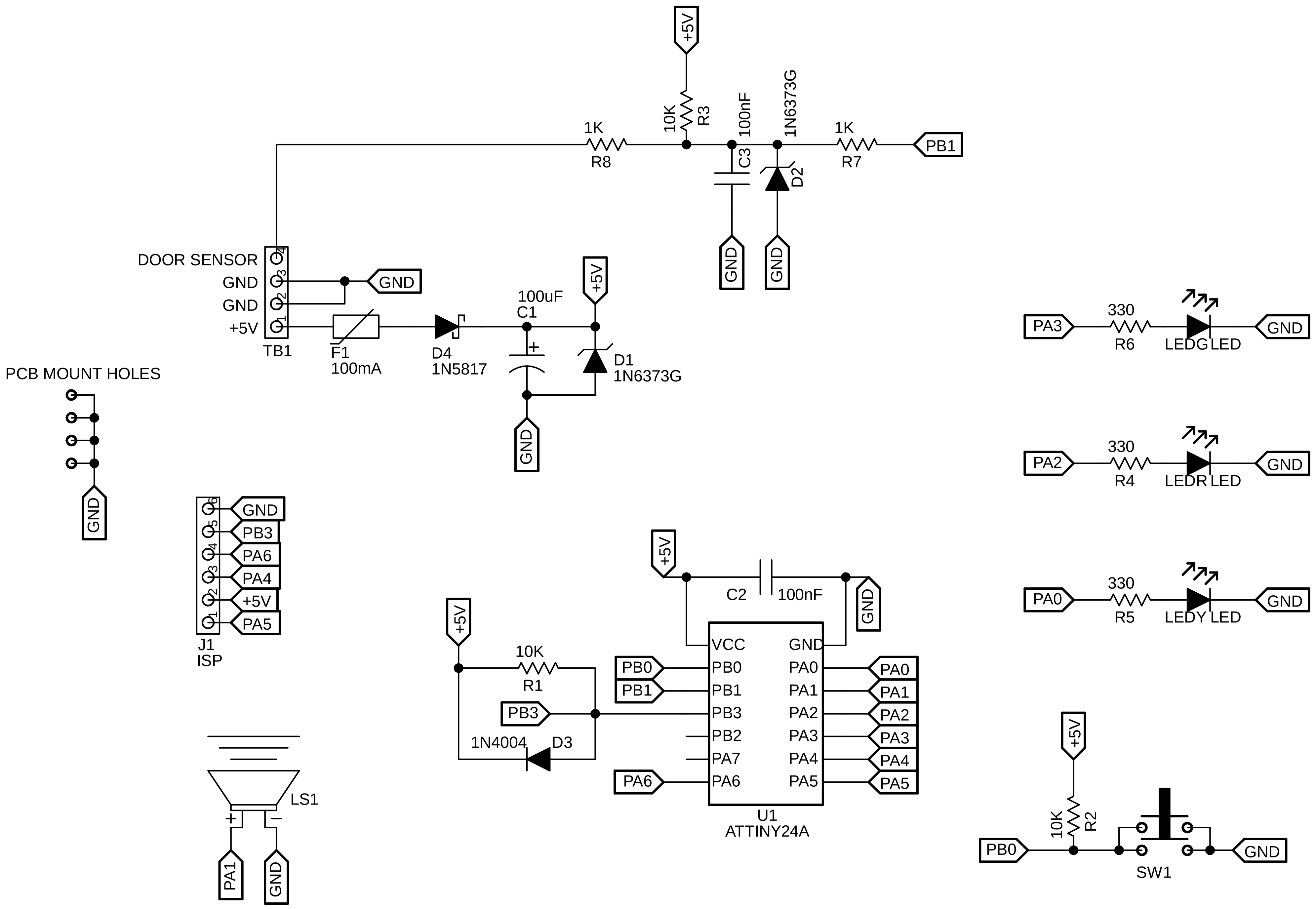
Output system: The beeper I am using doesn't have any feedback when turning on or off and it is around 20mA so I am running it directly off a uC pin. The three LED's are also driven directly from the uC with current limiting resistors.

Input system: The disable alarm button has a pull-up resistor on although the code also uses the internal pin pull-up so it could likely be skipped. I have bulletproofed the garage door sensor more from ESD. I am using a standard type alarm system sensor which has its two conductors shorted when near its mating magnet. It goes through a resistor, ESD protection, smoothing capacitor, pull-up resistor, and then through another resistor to the uC pin.

Parts List:

Part	MPN	Digikey	Qty	Extended Price	Notes
C1	ECE-A1AKA101	P831-ND	1	0.25	
C2-C3	K104Z15Y5VF5TL2	BC1160CT-ND	2	0.88	
D1-D2	1N6373G	1N6373GOS-ND	2	1.24	
D3	1N4004	1N4004FSCT-ND	1	0.34	
D4	1N5817	1N5817FSCT-ND	1	0.54	
F1	MF-R010	MF-R010-ND	1	0.16	
LEDG	SSL-LX5093LGD	67-1108-ND	1	0.54	
LEDR	SSL-LX5093LID	67-1110-ND	1	0.39	
LEDY	SSL-LX5093LYD	67-1111-ND	1	0.39	
LS1	CEM-1205C	102-1124-ND	1	2.73	
R1-R3	ERO-S2PHF1002	P10.0KCACT-ND	3	0.45	
R4-R6	ERO-S2PHF3300	P330CACT-ND	3	0.45	
R7-R8	ERO-S2PHF1001	P1.00KCACT-ND	2	0.30	
SW1	FSM14JH	450-1642-ND	1	0.09	
TB1	1725672	277-1275-ND	1	2.23	
U1	ATTINY24A-PU	ATTINY24A-PU-ND	1	2.01	
U1-SOCKET	4814-3004-CP	3M5474-ND	1	0.18	
PWRSUP	EPS050100-P5RP	T1038-P5RP-ND	1	6.09	Power Adapter
BOX	110I,AL	SR110-IA-ND	1	3.56	Almond Box
BOX	110I,BK	SR110-IB-ND	1	3.56	Black Box
BOX	110I,GY	SR110-IG-ND	1	3.56	Gray Box


Garage Monitor
Closed
Opened
Disable Beeps



+5V
10K
R3
100nF
C3
1N6373G
D2
1K
R8
1K
R7
PB1
GND
GND
DOOR SENSOR
GND
GND
+5V
TB1
GND
F1
100mA
D4
1N5817
100uF
C1
+5V
D1
1N6373G
GND
PCB MOUNT HOLES
GND
PA3
330
R6
LEDG LED
GND
PA2
330
R4
LEDR LED
GND
PA0
330
R5
LEDY LED
GND
J1
ISP
GND
PB3
PA6
PA4
+5V
PA5
+5V
C2
100nF
GND
+5V
10K
R1
PB3
1N4004
D3
VCC
PB0
PB1
PB3
PB2
PA7
PA6
GND
PA0
PA1
PA2
PA3
PA4
PA5
U1
ATTINY24A
PB0
PB1
PA6
PA0
PA1
PA2
PA3
PA4
PA5
LS1
PA1
GND
+5V
10K
R2
PB0
SW1
GND