Automobile Wireless Door Lock Controller

Recently I bought a pickup truck that did not have wireless remote control locks or even electric locks. I found that I really missed that option. I purchased an after market remote electric lock kit and it worked fine. However, it is a very basic system and does not do some of the things that factory remote locks do. I decided to build my own controller to replace the one in the kit and make it do everything I wanted.

I decided to base the controller on an Arduino UNO and a four button 433mhz wireless remote control. It did not take long to get the controller working on the bench and I started testing it with some lock motors. Everything worked fine until I measured the idle current. It took about 50ma doing nothing. That led me to learn how to reduce the power used by the Arduino.

I changed the processor board from the UNO to the Arduino Pro Mini. This is an almost minimum arduino that is compatible with the UNO. After the prototype was changed to the Pro Mini the idle current was reduced to about 25ma. Removing the LED pilot light on the Pro Mini the idle current was about 22ma.

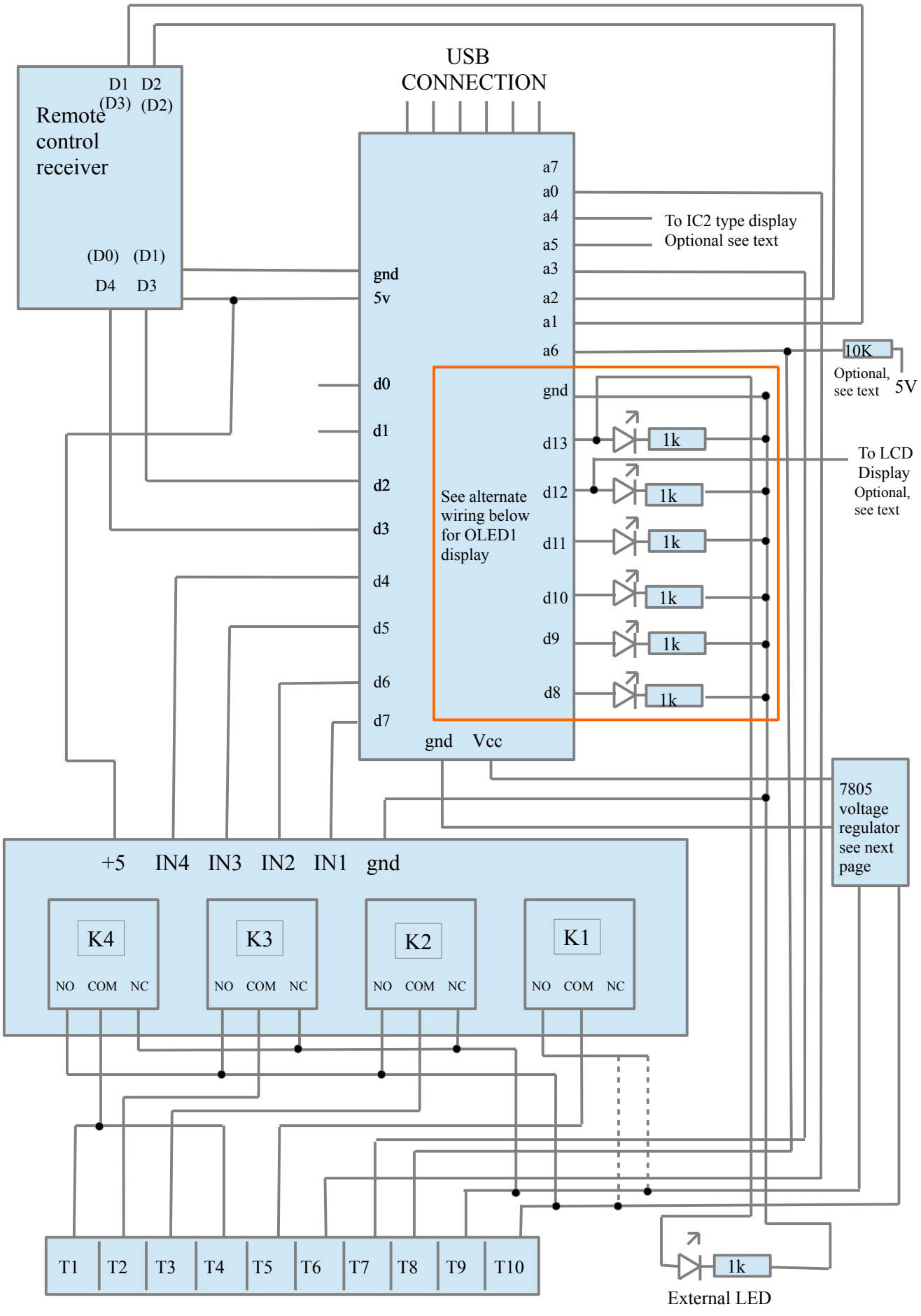
I still wanted it to be much lower. It is possible to put the processor on the Pro Mini to sleep and after In my final implementation the processor goes to sleep after about 1 second of no commands from the remote, but the remote control receiver must stay alive. When any command is received the radio will wake the processor to process the command.

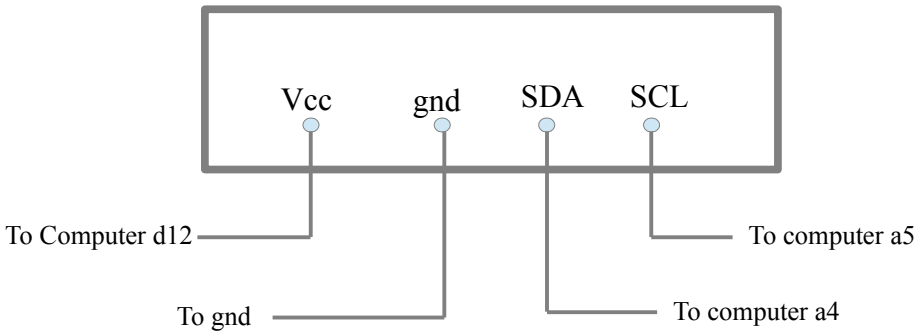
With this done the idle current is about 10ma. I can live with that!

The following is a description of how all this works. Starting with a schematic of the connections to the Pro Mini.

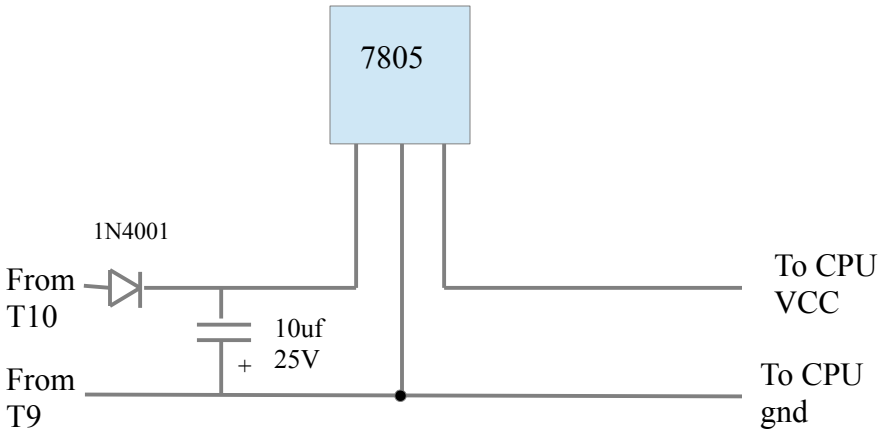
If you want a copy of the Arduino sketch directory just point you browser to:
http://firebottleradio.com/watts/lock.ino

The source code will be displayed. Just save this to a directory named “lock” in your Arduino sketch directory.

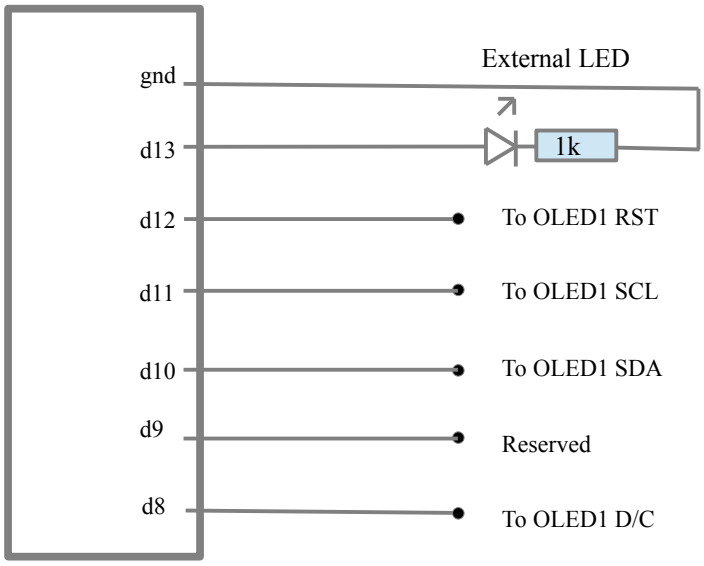

USB
CONNECTION
Remote
control
receiver
D1 D2
(D3) (D2)
(D0) (D1)
D4 D3
gnd
5v
a7
a0
a4
a5
a3
a2
a1
a6
To IC2 type display
Optional see text
10K
Optional,
see text
5V
d0
d1
d2
d3
d4
d5
d6
d7
See alternate
wiring below
for OLED1
display
gnd
d13
d12
d11
d10
d9
d8
1k
To LCD
Display
Optional,
see text
gnd Vcc
7805
voltage
regulator
see next
page
+5 IN4 IN3 IN2 IN1 gnd
K4
K3
K2
K1
NO COM NC
T1 T2 T3 T4 T5 T6 T7 T8 T9 T10
External LED



I2C type display
(SAIN, LCD, LCD2,
DIGOLE)



Voltage Regulator



Alternate wiring when OLED1
Display is used
(Remove the 6 LEDs)

Parts List

Part	Supplier	Part No.	Cost
FTDI Friend	Adafruit	ID# 284	$14.50 Note 1
Relay board	Sainsmart	SKU:20-018-101	$6.99 Note 2
4 channel remote	RF Wireless Electric	4 ch rolling code	$16.14 Note 3
LED board	RF wireless electric	6 led plug in	$0.99 Note 4
Lock motors	A1 Electric	W16F	$19.95 Note 5
Terminal strip	Radio Shack		$3.99
perf board	Radio Shack		$3.99
voltage regulator	Radio Shack	276-1170	$1.95
Pro Mini board	eBay		$2-$6
LCD Shield	Adafruit	ID# 772	$19.95 Note 6
Diode	Radio Shack	1N4001	$1
Capacitor	Radio Shack	10uf 25vdc	$1
OLED display	eBay	Note 7	$4-$12 Note 6

Note 1:You will need at lease one of Adafruit's FTDI boards to program the Pro Mini. If you do another project you can reuse the FTDI board.

Note 2: A number of suppliers on eBay sell this board or one exactly like it.

Note 3: eBay store: RF Wireless Electric (eBay seller ID canton-electronic) sells a lot of RF stuff. I chose the 4 channel rolling code controller and receiver. If you select another one the names of the output signals may be different.

Note 4: eBay store: RF Wireless Electric (eBay seller ID canton-electronic) has several items they call rapid prototype. I used their little 6 led board that plugs directly into the Arduino on pins 8-13 and gnd.

Note 5: You will need two two of these for a two door pickup. A1 sells the very best for $19.95, but you can get a good quality motor for a lot less on eBay. The A1 motor uses about 3 amps and the low cost ones use about 4 amps. The A1 W16F is a five wire motor. You will need a five wire if you want some of the controller options to work (see below). If you don't want those options you can use a lower cost 2 wire motor.

Note 6: The software is written to use three different configuration methods. (1) the six LEDS on the controller and (2) a 16 character by 2 line LCD display. (3) OLED display. TheLED method is much less expensive, and the LCD display is more expensive but much easier to use. The OLED display is as easy to use as the LCD and less expensive. It is also very small... only 2mm thick. Also, the standby power for the OLED is virtually zero (it has a sleep mode). Because of this I could use the OLED for status as well as configuration and it can replace the LEDs as well. I could not do this with the LCD because it did not like being turned off and on without a computer reboot. All things considered I think the OLED is the way to go.

#define LCD
for LCD support or:

#define LEDCONFIG
for configuration by LEDs

#define OLED1

for OLED display

Note 7: There are dozens of OLED displays available. Get one that is supported by the U8G library (https://code.google.com/p/u8glib/). You will have to work out the wiring for the specific one that you use. I used SPI but there are UART and I2C versions available as well. I have an I2C on order and will give it a try shortly.

Packaging:

I mounted the Pro Mini on a piece of Radio Shack perfboard. When I built the Pro Mini I used long male headers instead of the short male headers provided with the kit. It is soldered into the perfboard I cut and drilled the perfboard so that it would mount on top of the relay board with 1" standoffs. Female headers on the perfboard accepted the remote control receiver. Everything was wired with wire wrap. Old technology, but easy to do and easy to change if required. The whole thing was mounted in a small plastic project box.

This Pro Mini has an on board 5v regulator, but it is TINY and gets pretty warm when fed with 14v (car battery when charging) so I decided to bypass the on board regulator and put a 7805 on my perf board. It has a small heat sink and if necessary I can add a little more heat sink.

Wireless Remote Control:

The remote control has 4 buttons. I used the top left as LOCK and the top right as UNLOCK. If configured, LOCK will beep the horn one time and UNLOCK two times. The bottom left button will LOCK but does not beep the horn. The bottom right will UNLOCK with out horn. These buttons are referred to as QUIET LOCK and QUIET UNLOCK.

LCD Display:

There are two options for configuration. (1) Use the 6 LEDs on the controller and (2) use a 16x2 line LCD display. The LEDs option cost almost nothing, but is a little hard to use. The LCD option cost about $20 but is VERY easy to use. You must select the option when compiling the sketch. See Note 6 above for that selection. It is not recommended to have LED and LCD at the same time.

When the LCD option is selected the LCD will be powered down and not enabled in the software until the CONFIG mode is enabled (Quiet Lock and Quiet Unlock at the same time). When the CONFIG mode is exited (Quiet Lock and Quite Unlock at the same time) the LCD is turned off and the computer is rebooted. The LCD requires almost 8 ma with the back light on and 3 ma with the back light off. This is too much current to leave it on all the time, even with the back light off.

OLED Display:

There is now a third (UPDATE: really 5, see below) option for configuration and status display. The software now supports an Organic Light Emitting Diode (OLED) display. The OLED replaces both the 6 LEDs and the LCD display. I used a neat little OLED that has a SH1106 controller and it interfaced via SPI. It is driven by the U8GLIB library (https://code.google.com/p/u8glib/). If you want to use this little display download and install the U8GLIB and add: #define OLED1 to the sketch. Remove the other #defines that select display options. The software is set up to use the 5 of the I/O lines that were

used to drive the LEDS . D8 is connected to D/C on the OLED. D10 is connected to SDA on the OLED. D11 is connected to SCL on the OLED. D12 ss connected to RST on the OLED. The constructor for the U8G object wants a CS line but the OLED that I used did not use this line. Because of this I had to assign D9 to CS even though it is not used. If you use some other OLED you will have to sort out the pins and controller type and make changes to the constructor.

Since the OLED that I selected supports a sleep mode the current when asleep is 9.7ma and when awake is 26ma for the entire controller.

When the controller wakes up it uses the OLED to show the status. When in config mode the display is the same as with the LCD display.

To wire this display just remove the little 6 LED board and use that header to connect the wires from the OLED.

OK, I admit it... Displays for this project have become somewhat of an obsession. I have implemented five (six if you count the 6 LEDS) different display types. They are selected by #define in the software. Just remove the comment from the line with "USEME" to enable an option.

Here is the configuration from the code:

```
//enable LEDCONFIG to use 6 leds for configuration
//#define LEDCONFIG USEME

//enable LEDSTAT to use 4 of the leds for operating status (LOCK, UNLOCK,
//         2ndUnlock, HORN)
//#define LEDSTAT USEME

//enable LCD to use the LCD for configuration. If LCD is enabled LEDCONFIG must
//       be disabled and LEDSTAT must be enabled since the LCD does not provide operate status and the
//       relays are operated from the status procedure
//     LCD is the LCD display from Adafruit
//#define LCD USEME

//enable OLED1  to use the OLED display type 1. If OLED1 is enabled LEDSTAT and
//      LEDCONFIG MUST be disabled because (they use the same IO lines)
//     OLED1 is the OLED display from Upgrade Industries
#define OLED1 USEME

//enable DIGOLE to use the Digole LCD for configuration. If DIGOLE is enabled
//      LEDCONFIG must be disabled and LEDSTAT must be enabled since the DIGOLE does not
//      provide operate status and the relays are operated from the status procedure
//#define DIGOLE USEME

//enable LCD2 to use the LCD (type 2)for configuration. If LCD2 is enabled
//      LEDCONFIG must be disabled
//     LCD2 is the LCD display from YWOrobot
#define LCD2 off
//#define LCD2 USEME

//enable SAIN to use the Sainsmart OLEDfor configuration and status. If SAIN is
//      enabled LEDCONFIG  and LEDSTAT must be disabled
```

// SAIN is the OLED display from SainSmart IIC OLED 1.04_OLED V.1.
//#define SAIN USEME

There is a conditional code “trap” that should catch any invalid configuration.

All of these displays are I2C except for OLED1. OLED1 uses 5 IO pins (12,11,10,9,8). Each one of these displays uses a different library. See the suppliers information to get the correct library.

Strange use of A6:

You will notice in the schematic that A6 is wired differently from the other analog pins used as digital input. On the Pro Mini A6 and A7 are analog ONLY. They can not be used as digital like A0-A5. With the addition of a pull up resistor A6 can still be used to sense an external switch that grounds the input line. There is still a problem with this... A6 and A7 will not trigger the Pin Change interrupt and can not wake up the computer. Therefore they can not be used for SWITCH1 or SWITCH2. Fortunately, RELOCK/LOCKOUT is only needed when the computer is awake since the software keeps the computer awake while it times these function out. **REVISED**: I noticed that on my truck (Toyota Tacoma) there was a small voltage on the door switch when the door was closed. This may be from a pullup in the system that senses the door open/closed. I removed the 10K pullup on A6 and it still works. I feel better about this. Avoids dueling pull ups. **ANOTHER REVISION:** I just discovered that my Toyota Tacoma does NOT like to have the door switches connected to the input of the Arduino A/D. I would have thought that the input impedance of the A/D would be high enough that the other things that use the door switch would not know that it is there. However, the dome light works correctly and the door open icon on the instrument panel works correctly, but the beeper that warns you that the key is in the switch when you open the door does not work. It always thinks the door is open!! I will have to think of another way to do this function.

Connections (refer to the schematic):

K1-K4 are 10 amp relays on a four relay board. These boards are available on eBay for less than $10, but be sure you get 10 amp relays. The board will have opto isolators and relay driver transistors so they can be connected directly to the processor output pins.

T1-T10 is a terminal strip to make all the external connections. You can make connections from the terminal strip to the processor with 20 or even 22 awg wire, but the wires to the contacts of the relays should be at least 20 or even 16 awg since the lock motor current goes through these wires.

T1 is the driver side lock motor terminal A See Note 1
T2 is the driver side lock motor terminal B See Note 1
T3 is the passenger side lock motor terminal A See Note 1
T4 is the passenger side lock motor terminal B See Note 1
T5 is the horn control. See Note 2
T6 is Switch 1 connection. See Note 3
T7 is Switch 2 connection. See Note 3
T8 is the LOCKOUT/RELOCK switch connection. See Note 4
T9 is the the automobile chassis ground. See note 5
T10 is the automobile battery positive. See Note 5

Note 1: If you use two wire motors then this will be one of the two wires. You will have to experiment to see which is A and which is B. If you have them backward then LOCK and UNLOCK will be reversed.

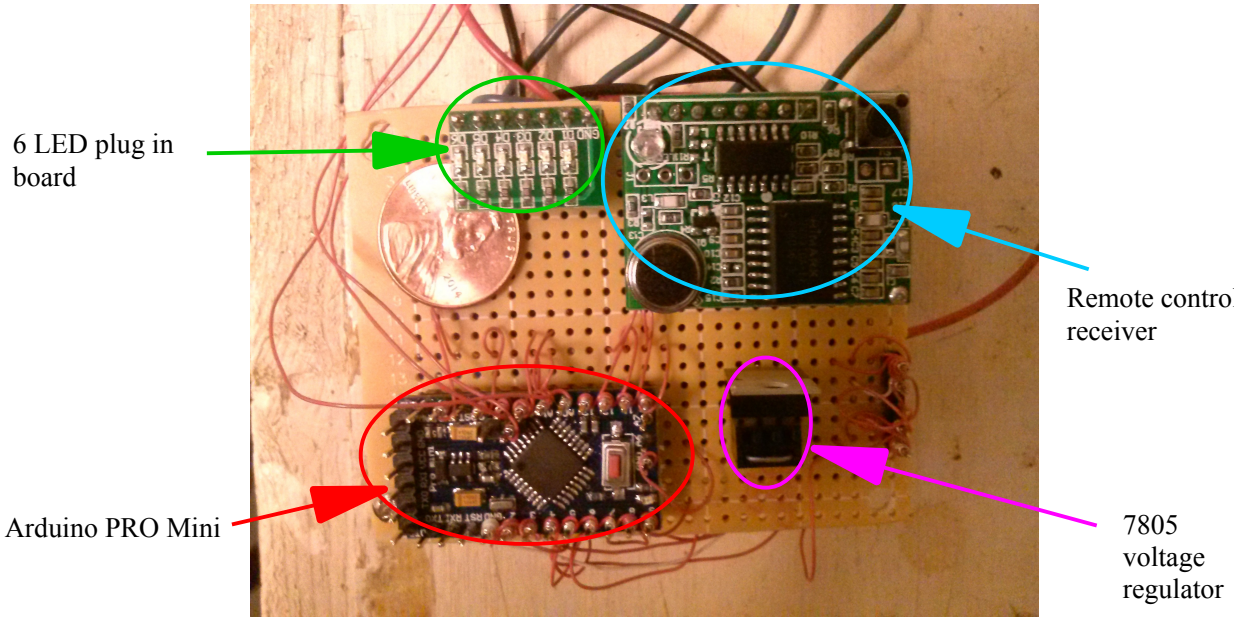
Note 2: Notice that the connection to the NO contact of the horn relay is dotted. Connect NO to either +12 volts or ground depending upon whether you need +12 or ground to blow the horn.

Note 3: T6 and T7 are used to sense the position of the lock motors. If you don't have 5 wire lock motors then you can't use this option. T6 should be connected to the driver side motor wire that is grounded when that door is locked. T7 should be connected to the passenger side motor wire that is grounded when that door is locked.

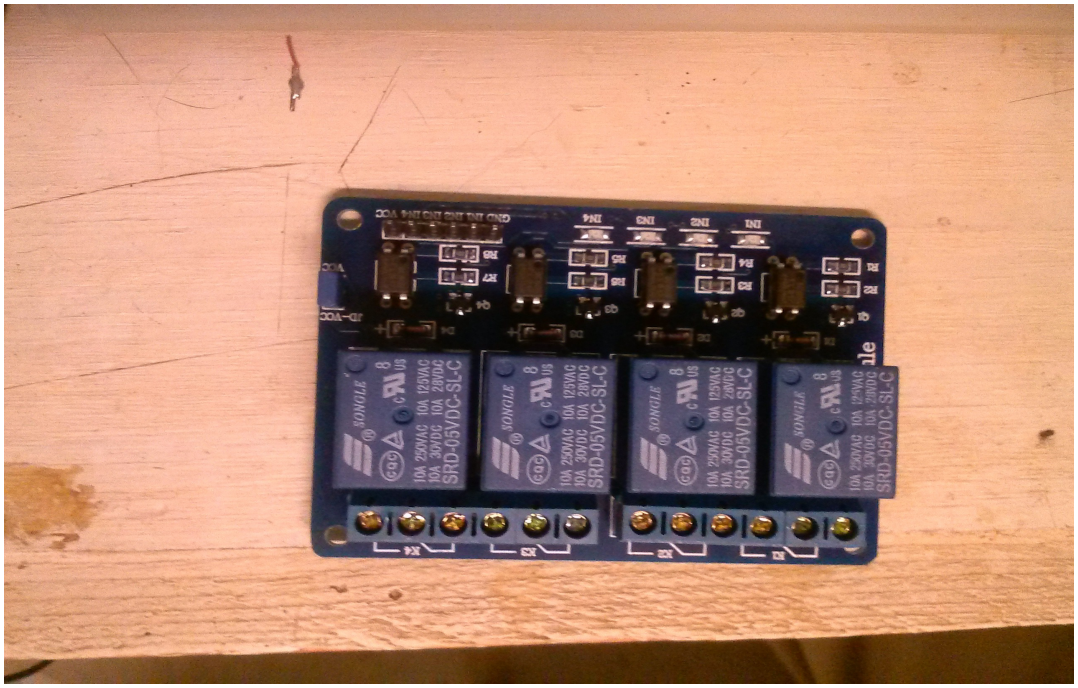
Note 4: T8 is used to sense that a door has been opened. If either of these functions are enabled this line must be connection to the door sense button in the door frame that grounds the line when the door is open. This line controls two independently configurable options in the controller. (1)RELOCK – if the door is unlocked remotely but not opened for 45 seconds it will be relocked. This protects from a spurious unlock or if you unlock the door and don't open it but forget to relock it. (2)LOCKOUT – if the door is open the remote can not be used to lock the door. This prevents locking the door, getting out and closing it when you may have left the keys and fob in the car.

Note 5: T9 must be connected to chassis ground with at least 16 awg wire. T10 must be connected to the battery positive (constant power supply) with at least 16 awg wire with a 15 amp fuse.

This system was designed for a two door pickup truck. If you want to use it on a 4 (or more) door car you will have to add additional relays and probably use larger wire in the relay 12V and ground wiring. The control lines from the processor can be paralleled. The door motors require about 3 amps (some as much as 4 amps) to operate. If you try to control three doors from one passenger relay you will probably exceed the current rating of the relays. Another option would be to find (or make) a relay board that has higher rated relays.



Controller Board



Relay Board

OLED display from Upgrade Industries

OLED display in the encolsure

Comments:

The following is a copy of the comments from the C code that runs the controller (don't depend entirely on this copy. Read the comments in the code as well):

Auto door lock controller UNO/duemilanove 328 Version
Works with Arduino Pro Mini (select duemilanove 328 in IDE)

this controller will control after market door locks. It will work with either two wire or 5 wire lock motors. It supports several modes of operation. These modes are configurable.

Leds and switches:

There is one external LED that shows the state of the controller.
Slow flash = UNLOCKED
Fast flash = LOCKED
Double fast flash = driver door unlocked waiting 15 seconds for second unlock command to unlock passenger door(s)
Very fast flash = entering or leaving configuration mode. If continuous there is a fault

Configuration:
Pressing remote Quiet LOCK and Quiet UNLOCK buttons at the same time will enable the configuration mode. If the LED configuration option has been selected at compile (#define LEDCONFIG) then the 6 LEDs will be used for configuration. When the configuration mode is enabled the external LED will flash rapidly for a few seconds then the six internal LEDs will display the software version. The version number is two HEX characters. The two left most leds show the first character (0-3) and the right 4 leds show the second character (0-F). e.g. version 24 would be: 100100 or hex 24. If the LCD display is fitted the version will be displayed in decimal. In this example it would be 36.

If the LCD configuration option has been selected at compile (#define LCD) then the LCD display will be used for configuration. The LCD configuration is controlled exactly the same as the LED mode, (LOCK steps throught the options, UNLOCK selects the option, Quiet LOCK and Quiet UNLOCK at the same time exits config.

There are 10 configuration items. The first 8 are true or false (enabled or disabled).
1: HORNENABLE: horn enable, the horn is beeped one time for lock and two for unlock
2: PROGRRESSIVE: progressive mode, one unlock will unlock driver door, second unlock (within 15 seconds) will unlock passenger door
3: TWO STATE SWITCHES: two state switch mode, SPDT center off momentary switches can be wired to input lines SWITCH1 and SWITCH2 so that they will ground the input lines when moved from the center off position. SWITCH1 will unlock and SWITCH2 will lock. IO lines should be wired to a switch on the driver door in parallel to a switch on the passenger door.
4: STAT SWITCHES: status switches are used. In this mode a microswitch is attached to the control inside the door so that if the manual (mechanical) lock or unlock lever is used on either door the controller will lock or unlock the other door. The input lines SWITCH1 and SWITCH2 are connected to the door switches, 1 to driver side and 2 to passenger side. This mode replaces the FIVEWIRE and TWOFIVEWIRE modes previously available. One grounding output on each 5 wire motor is used. The driver side line that is grounded when locked is connected to SWITCH1

and the passenger side line that is grounded when locked is connected to SWITCH2. No connection is made to the other statis lines from either motor.

5: LOCKOUT: The LOCKOUT input pin must be connected to a switch that will ground this input when a door is open. If a door is open the controller will NOT lock the doors. Press the UNLOCKCTRL switch or the remote UNLOCK button to change the state of this configuration item.

6: RELOCK: The RELOCK input line must be connected to a switch that will ground this input when a door is open (exactly the same as LOCKOUT mode). If the remote control is used to unlock the doors and no door is opened for 45 seconds then the doors will be relocked. **While this configuration and LOCKOUT both use the same input pin they are completely independent and either or both may be enabled or disabled.** Press the UNLOCKCTRL switch or the remote UNLOCK button to change the state of this configuration item.

7: SLEEP: The controller will go to sleep after a period of inactivity (see below for turther information about sleep). NOTE: if this item is disabled then the following item will be disabled also.

FOR LED CONFIGURATION MODE:

8: TIMER: The controller will go to sleep after a period of inactivity (see below for turther information about sleep), but will wake up every 8 seconds and run for 1 seconds. This will allow the external LED to show the locked or unlocked state and to show that the controller is alive.

FOR LCD or OLED CONFIGURATION MODE:

8: AWAKE TIME and TIMER enable: Press remote UNLOCK to select the time (ms) between 0 and 1000 ms that the computer stays awake. Select 0 to disable the wake up timer

9: LONGSLEEP: This option will enable and LONG sleep. If TIMER is enabled this option will add an additonal 5 seconds to the sleep time. If TIMER is disabled this option has no effect.

NOTE: if the TIMER is enabled then then SLEEP will be enabled also (can't do the timer thing without sleep). With this option enabled the average idle current is about 11.4ma. With this item disabled the average current is about 9.7ma.

Pressing the remote LOCK button will step through the above 8 configuration items one at a time. The left most LED will show the state of that item. The right 4 LEDs will show the number (in HEX) of the item. e.g. on the first press LED 1 will light and the left most LED (#6) will be ON if horn is enabled.

Pressing the remote UNLOCK button will change the state of this configuration item.

Each press of remote LOCK button will advance to the next configuration of the 10 items.

The 9th press of the remote LOCK button The controller can be configured for 200, 400, 600, or 800 msec duration of the motor pulse. When in this configuration mode LED1 and LED2 will be on and one of the other 4 LEDs will be on. LED3=200ms, LED4=400ms, LED5=600ms and LED6=800ms. Pressing the remote UNLOCK button will step through these 4 options. The shortest motor pulse that operates the locks reliable should be used.

The 10th press of the remote LOCK button The controller can be configured for 50, 100, or 200 msec duration of the horn pulse. When in this configuration mode LED1, LED2, and LED3 will be on and one of the other 3 LEDS will be on. LED4=50ms, LED5=100ms and LED6=200ms. Pressing the remote UNLOCK button will step through these 3 options. Pick the pulse that gives the best horn beep.

The 11th press of the LOCKCTRL button brings the configuration back to the first press mode.

When the remote Quiet LOCK and the remote Quiet UNLOCK buttons are pressed at the same time the

external LED will flash rapidly for a few seconds and the configuration will be stored in NV RAM and the controller will return to normal operation.

If no buttons are pressed while in config mode for 30 seconds then config mode will be exited. The config WILL NOT BE STORED in NV RAM and the processor will be rebooted to restore the config as it was before this config operation was started.

If the horn beep is enabled it will sound one beep when the door is locked by the remote control button and two beeps when the door is unlocked by the remote control button. If the remote quiet lock button is pressed the door will be locked but there will be no horn beep. Likewise, if the quiet unlock button is presed the door will be unlocked without horn beep.

If more than 7 consecutive lock or unlock commands occur without a pause longer than 100ms the system will be disabled and the PILOT led will flash rapidly and continuously. This can be cause by a stuck switch on one of the 5 wire motors. The only way out of this fail safe mode is to restart the controller by removing the power and restoring it OR by pressing the LOCK and QUIET LOCK buttons at the same time. In order to avoid excessive battery drain the processor will sleep for 8 seconds and wake up to flash the LED for 1 second.

The switches used in the TWOSTATESWITCH mode must be single pole double throw center off type switches. The switches must NOT remain in either LOCK or UNLOCK mode for more than 5 seconds. If one of the switches does remain operated for 5 seconds that switch will be ignored until it is returned to the center off position. The remote control will continue to work while the switch is stuck.

If a button is stuck on a remote or if the receiver thinks that a button is stuck the controller will wait 5 seconds for the signal to clear. If it does not clear then that button signal will be ignored until it clears. While the button is ignored the other buttons will continue to work.

During normal operation the 6 leds will indicate the actions performed by the controller.

LED1 will flash when the LOCK relay is operated.
LED2 will flash when the drivers door UNLOCK relay is operated.
LED3 will flash when the passenger door UNLOCK relay is operated.
LED4 will flash when the HORN relay is operated.
LED5 is not used during normal operation.
LED6 will flash the same as the external PILOT light LED.

In order to conserve battery power the controller will go to sleep after about 1 second of inactivity. It can wake up when any remote button is pressed or, if one of the switch modes is enabled, when one of the door motors changes state (due to manual operation of the lock). Sleep is delayed if progressive mode is enabled and the driver door is unlocked. Sleep will not occur until the 15 second wait for the second push has expired. Also, if RELOCK is enabled sleep will be delayed until the 45 second delay wait for a door to open has expired. After the timeout or after a door is opened the controller will go to sleep.

The PILOTLIGHT and LED6 on the internal status will NOT blink during sleep time.