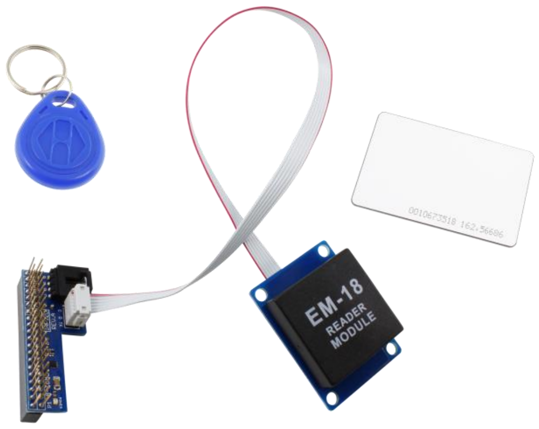
Using EM18 RFID Receiver for Raspberry Pi Zero, B+, 2 or 3

8/21/2016 – R.Grokett



ControlEverything (www.controleverything.com) offers a number of interesting products for Arduino, Particle, Raspberry and their own $5 Onion Omega 2 (Linux with WiFi computer). One item is the EM18 RFID reader module.

The EM18 RFID Receiver for Raspberry Pi uses I2C interface for communication.

RFID (Radio Frequency Identification) uses electromagnetic fields to read, monitor and transfer data from tags attached to different objects. It is not necessary that the cards are to be in visibility of the reader, it can be embedded in the tracked object. The tags can be actively powered from a power source or can be passively powered form the incoming electromagnetic fields.

EM-18 RFID reader module is one of the commonly used reader and can read any 125KHz tags. It features low cost, low power consumption, small form factor and easy to use.

This project lets you use a Raspberry Pi to act like an electronic door key & lock. You wave your RFID badge and the Raspi lights an LED, sounds a buzzer and engages a 120v/220v relay that could be connected to an electric lock or solenoid to unlock the door. After 5 seconds, the lock reengages and the LED and buzzer turn off. From inside the door, you can have a push button that unlocks the door for 5 seconds so you can leave without an RFID.

Parts

- Raspberry Pi Zero, B+, 2, or 3 running Raspian Jessie

- RFID module from ControlEverything which includes a Pi adapter and RFID key
  https://www.controleverything.com/content/RFID-Reader?sku=INPI2U_EM18_RFID
  You can substitute the Grove RFID instead. Be sure to set it to UART mode:
  https://www.seeedstudio.com/Grove-125KHz-RFID-Reader-p-1008.html
  http://wiki.seeedstudio.com/wiki/Grove_-_125KHz_RFID_Reader

- 5v to 120v/220v ControlEverything I2C relay
  https://www.controleverything.com/content/Relay-Controller?sku=PCA9536_I2CR_R11
  Or substitute a Grove Relay module
  https://www.seeedstudio.com/Grove-Relay-p-769.html
  http://wiki.seeedstudio.com/wiki/Grove_-_Relay

- 5V Grove Buzzer Module
  https://www.seeedstudio.com/Grove-Buzzer-p-768.html
  http://wiki.seeedstudio.com/wiki/Grove_-_Buzzer

- LED with 220 ohm resistor or Grove LED module
  https://www.seeedstudio.com/Grove-LED-p-767.html

- Momentary contact push button

- RFID tags (if more than the one included)
  https://www.seeedstudio.com/RFID-tag-combo-(125khz)-5-pcs-p-700.html

- Circuit board or breadboard
  or Grove base shield, cables and connectors
  https://www.seeedstudio.com/Grove-Universal-4-pin-connector-p-789.html

- Wire, box, etc.

Wiring

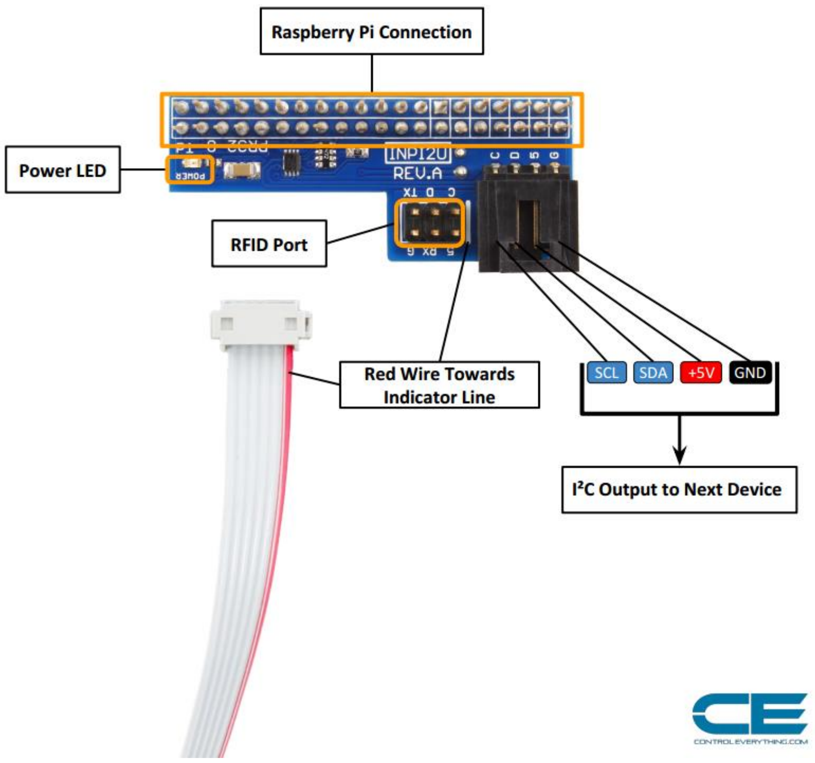
Using the ControlEverything EM18 RFID (or Grove) hardware, it is a straight forward task to connect everything together.

Note that on the Raspberry Zero, you will need to add a connector (i.e. solder!) so be aware. The other Pi's come with pin connectors already installed.

Note for Grove wiring, you will need to use Grove universal connectors and solder to a board as they no longer carry a Raspberry Pi shield.

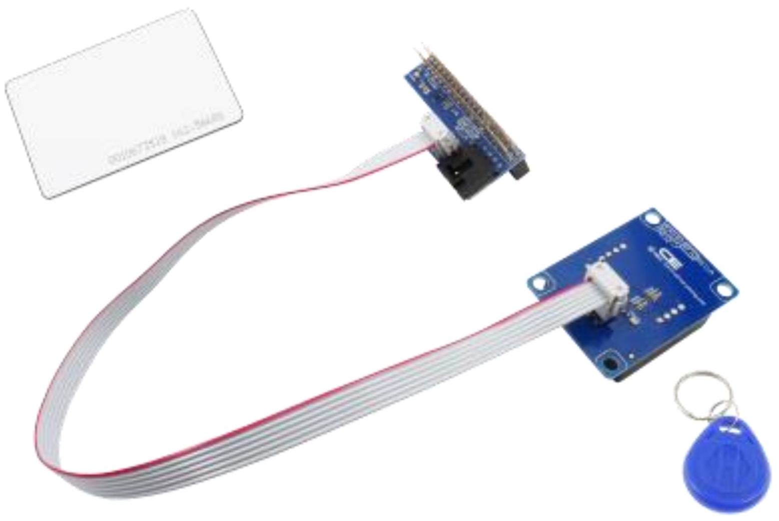
Take the ControlEverything Raspberry Pi I2C Adapter and line up with the P1 pin on the adapter facing the Pin 1 on the Raspberry Pi GPIO.

The RFID module uses the Raspberry's UART connections (GPIO 14, 15, +5v, Gnd). If you use the Grove RFID, set it to use the UART.   This is a Serial 9600 baud connection between the Raspi and the RFID module.



Next plug the included grey cable into the Raspberry Pi Adapter 6 pin port, making sure the red wire is facing the white indicator line on the adapter board.

Then plug the grey cable into the RFID module, again with the red wire facing the white indicator line.



Simple Start

Once you have plugged in your RFID module to the Pi, you can try it out.

Create a file called “rfid.py”

```
$ nano rfid.py
```

Edit the file using below.

You can also go to GITHUB https://github.com/rgrokett/RFIDReaderPi

```
import serial

# For PiB+ 2, Zero use:
ser = serial.Serial('/dev/ttyAMA0', 9600, timeout=1)

# For Pi3 use
#ser = serial.Serial('/dev/ttyS0', 9600, timeout=1)

while True:
    string = ser.read(12)

    if len(string) == 0:
        print "Please wave a tag"

    else:
        string = string[1:11]    # Strip header/trailer
        print "string",string
        if string == '650033A2E5':
            print "hello user 1"
```

Note that for Pi3, you need to comment out the Pi2 line and uncomment the Pi3 line as Pi3 uses /dev/ttyAMA0 for Bluetooth.

Also note that on the Pi3, you need to add the following line to /boot/config.txt:

```
$ sudo nano /boot/config.txt
```

Append:

```
#Enable UART
enable_uart=1
```

Then reboot your Pi3.

To run, just type:

```
$ sudo python ./rfid.py
```

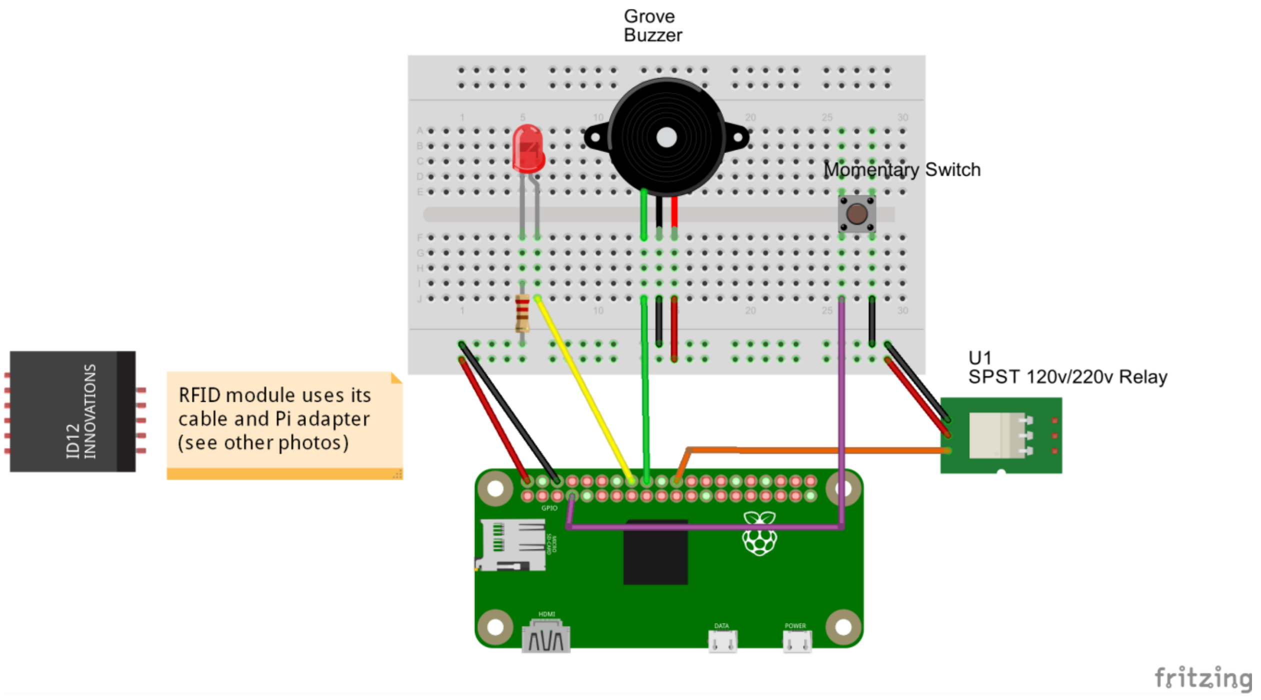
Tap your RFID card or fob against the EM18 module. The program should print out the ID string. You can edit the rfid.py to update the string with your ID's string.

Once this works, you can continue on to add the LED, Buzzer, Relay and button.

Part 2 Wiring

Use the diagrams and photos to assist in wiring the LED, Buzzer, Relay and button. Note that the Grove wiring will be slightly different as it uses +5v, GND and Signal leads, instead of just two wires.

If you are using the ConnectEverything I2C Relay module, you just plug it into the Pi Adapter I2C port.



For the other devices (LED, Buzzer, Switch and relay) connect using the diagram and pins shown.

Pin Layout

PIN	GPIO Pin	Device
2	5V	Breadboard 5v
6	GND	Breadboard Ground
7	GPIO 4	Momentary Button
16	GPIO 23	LED anode
18	GPIO 24	Buzzer (Signal lead)
22	GPIO 25	Relay (Signal lead)

Note for Grove wiring, you will need to use Grove universal connectors and solder to a board as they no longer carry a Raspberry Pi shield.

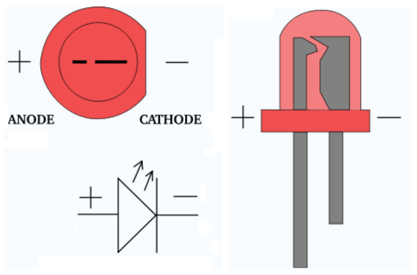


Fig. LED pins

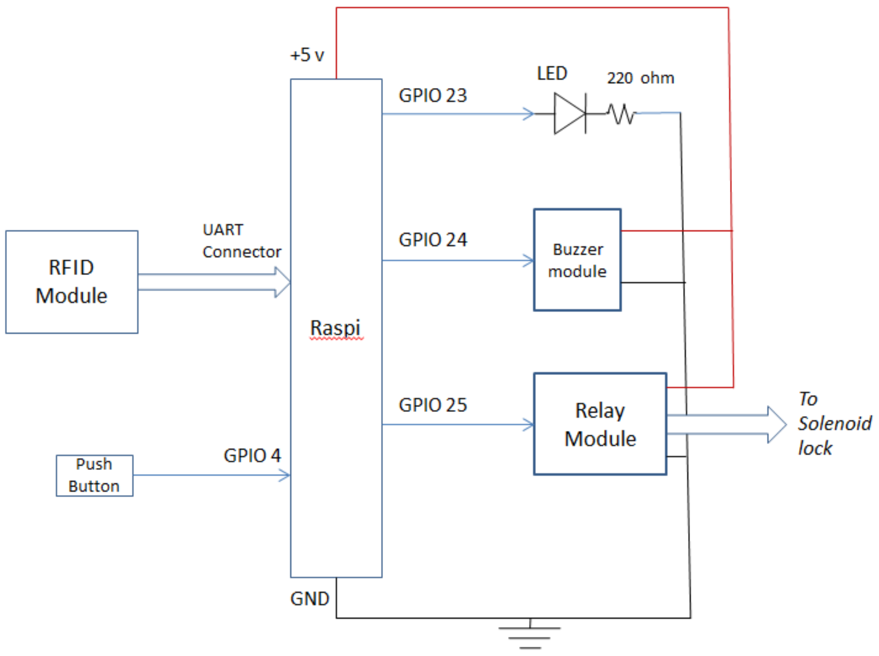
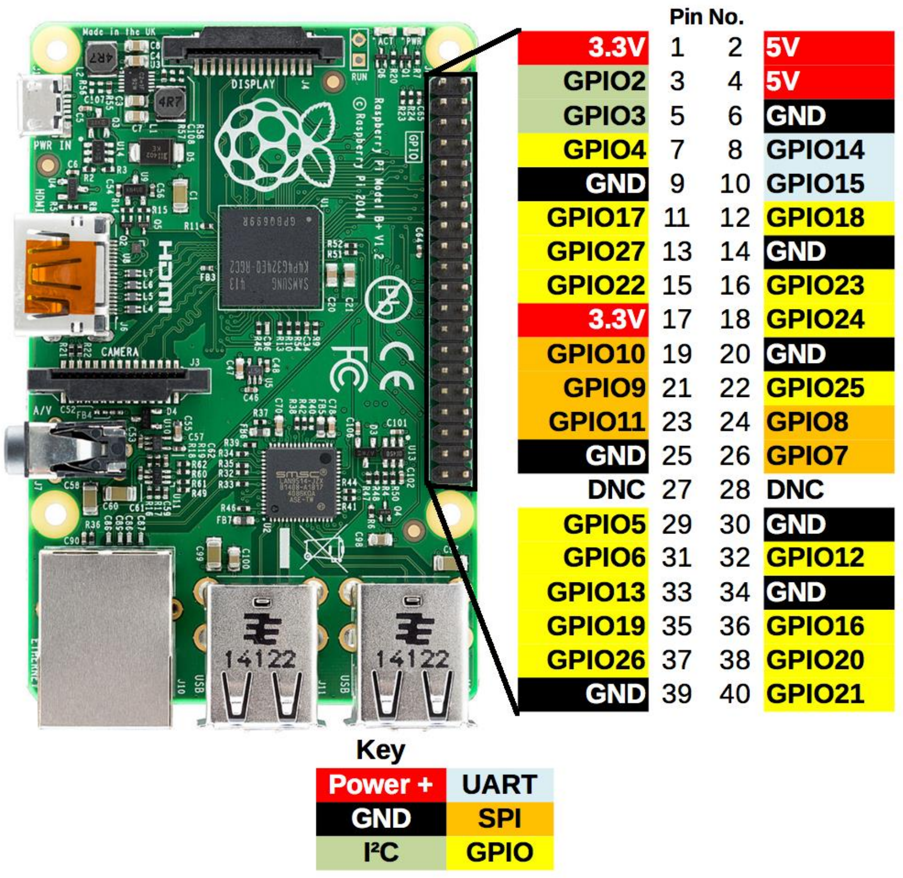


Fig. Wiring Diagram



	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

Key

Power +	UART
GND	SPI
I²C	GPIO

The GPIO pins for Raspberry Zero, B+, 2 and 3 are identical, so the same diagram will work for either.

When you initially turn on your Raspberry, the LED, buzzer and relay should stay off, but it is possible that they come on. Assuming no wiring errors, these will turn off when you first run the Reader program described below.

More Advanced Reader Software

There are two possible Relays supported here. The Grove version uses three wires (+5v, GND, GPIO Signal) while the ControlEverything uses I2C interface.

For I2C, you must activate the I2C interface on your Pi. This is not needed for Grove.

1. $ sudo raspi-config
2. Select "Advanced Options"
3. Select "I2C"
4. Answer "Yes" to Would you like the ARM I2C interface to be enabled?
5. Answer OK and select "Finish" to exit.
6. $ sudo reboot

After reboot, log back in and enter:

```
$ sudo i2cdetect -y 1
```

If you have no I2C devices plugged in, you will see the below.

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Once you plug in a device and rerun the command, you should see it show up in the list. (Probably need to reboot, if you didn't power down before plugging it in.)

Note that the EM18 RFID module isn't an I2C device, so doesn't show up.

The ControlEverything 1 or 2 channel relay module is I2C, so should show up in the I2C devices list as 0X41.

https://www.controleverything.com/content/Relay-Controller?sku=PCA9536_I2CR_R21

You can use the program below (or get the GITHUB version for the latest updates).

GITHUB https://github.com/rgrokett/RFIDReaderPi

```
$ git clone https://github.com/rgrokett/RFIDReaderPi.git
```

If you are using the I2C relay, you need to change the line

```
I2C = OFF          # Grove Relay
```

to

```
I2C = ON           # I2C Relay
```

If necessary, you can change which GPIO pins you are connecting your LED, Buzzer, Grove Relay and Push button as needed. But you cannot use the I2C or UART GPIO pins.

Add your RFID card numbers to the CARDS list in the program. If you don't have the number from the rfid.py program, any card you wave while running this program will be printed on the screen, but will not "open" the door.

Run the program using:

```
$ sudo python reader.py
```

The program will display a message every few seconds watching for a card to read. When it sees one, it will turn on the LED, Buzzer and Relay for 5 seconds, then turn them all off.

If you press the momentary button instead of waving a card, it too will "open" the door.

If you exit the program (Ctrl-C), the relay will engage to "unlock" the door on failure. (The Jurassic Park mode!) This would keep you from being locked out if the Raspi failed.

Note that I did not include error trapping or set up the program to restart on reboot. Also note that the relay will NOT be engaged if the power goes out. This means the door would be "locked"!

A Real System

For a real Door lock system, you need to add the electric lock or solenoid to the Relay. You also need to harden the program and the Pi for failures. (i.e. error trapping and restart at reboot, a battery backup, a case for the RFID module, etc.) If you have many RFID cards, an external file or database would be useful, along with the user's name.)

See also the version "reader2.py" in GITHUB for example of these extra features.

```
#!/usr/bin/python
#
# reader.py -- RFID READER
# version 0.9
# r.grokett
#
# If card is found, turns on LED,buzzer,relay
#
# Note that relay can be GPIO driven type or I2C driven type
# Both are called, even if not used.


import smbus
import time
import serial
import sys, signal, os, datetime
import RPi.GPIO as GPIO


# INITIALIZE
CARDS = [
        '1234567890',
        'E0043DBB52'
]

ON  = 1
OFF = 0

I2C = OFF       # Grove Relay  (Set to ON for I2C Relay)

# Set up GPIO Pins
LED     = 23
BUZZER  = 24
RELAY   = 25
BUTTON  = 4

GPIO.setmode(GPIO.BCM)
GPIO.setup(LED, GPIO.OUT)
GPIO.setup(BUZZER, GPIO.OUT)
GPIO.setup(RELAY, GPIO.OUT)
GPIO.setup(BUTTON, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Get I2C bus
bus = smbus.SMBus(1)



############################
# METHODS
############################
# TURN ON/OFF LED
def led(value):
    print "led()"
    if (value):
        GPIO.output(LED, GPIO.HIGH)
        print "LED state is : HIGH"
    else:
        GPIO.output(LED, GPIO.LOW)
        print "LED state is : LOW"

# TURN ON/OFF BUZZER
def buzzer(value):
    print "buzzer()"
    if (value):
        GPIO.output(BUZZER, GPIO.HIGH)
        print "BUZZER state is : HIGH"
    else:
        GPIO.output(BUZZER, GPIO.LOW)
        print "BUZZER state is : LOW"
```

```
# TURN ON/OFF GPIO RELAY
def relay(value):
    print "relay()"
    if (value):
        GPIO.output(RELAY, GPIO.HIGH)
        print "RELAY state is : HIGH"
    else:
        GPIO.output(RELAY, GPIO.LOW)
        print "RELAY state is : LOW"

# TURN ON/OFF I2C RELAY
def i2c_relay(value):
    print "i2c_relay()"
    if not I2C:
        return

    if (value):
        # PCA9536_R11 address, 0x41(65)
        # Select output port register, 0x01(01)
        bus.write_byte_data(0x41, 0x01, 0x01)
        print "Relay-1 state is : HIGH"
    else:
        # PCA9536_R11 address, 0x41(65)
        # Select output port register, 0x01(01)
        bus.write_byte_data(0x41, 0x01, 0x00)
        print "Relay-1 state is : LOW"

# UNLOCK DOOR
def unlock_door():
    print "unlock_door()"
    led(ON)
    buzzer(ON)
    relay(ON)
    i2c_relay(ON)

# LOCK DOOR
def lock_door():
    print "lock_door()"
    led(OFF)
    buzzer(OFF)
    relay(OFF)
    i2c_relay(OFF)


# EXIT
def signal_handler(signal, frame):
  print "Exiting!"
  relay(ON)    # NOTE! UNLOCKS DOOR ON FAILURE/EXIT (Jurassic Park!)
  i2c_relay(ON)
  GPIO.cleanup()
  ser.close()
  sys.exit(0)


############################
# MAIN
############################
```

```
if __name__ == '__main__':

    signal.signal(signal.SIGINT, signal_handler)

    # Select configuration register, 0x03(03)
    # 0x00(00)  Set all pins as OUTPUT
    if (I2C):
        bus.write_byte_data(0x41, 0x03, 0x00)
    time.sleep(0.5)

    ser = 0
    # Selects which version of Raspi you have
    if (GPIO.RPI_REVISION == 3):
        # For Pi3 use
        print "This is a Pi 3"
        ser = serial.Serial('/dev/ttyS0', 9600, timeout=1)
    else:
        # For Pi2, B+, Zero use
        print "This is a Pi B+, 2, or Zero"
        ser = serial.Serial('/dev/ttyAMA0', 9600, timeout=1)

    # LOCK THE DOOR ON BOOT
    lock_door()

    while True:
        buffer = ser.read(12)

        if len(buffer) == 0:
            print "Please wave a tag"
        else:
            card_id = buffer[1:11]      # Strip header/trailer
            print "Card ID : ", card_id
            if (card_id in CARDS):
                print "FOUND CARD ID: ", card_id
                unlock_door()
                time.sleep(5)
                lock_door()
            else:
                print "CARD NOT IN LIST!"

        # EXIT THE BUILDING
        # Add an indoor button
        if not GPIO.input(BUTTON):
            print "Button Pressed"
            unlock_door()
            time.sleep(5)
            lock_door()

        time.sleep(0.5)
    # END WHILE
# END MAIN
```

Reader2.py Installation

The reader2.py is a more elaborate version of the door lock controller that stores cards in a file and logs date/time of entries and exits.

To use this version, copy the cards.dat to your /home/pi directory and edit with your card numbers.

```
$ cp cards.dat /home/pi/cards.dat
```

```
$ nano /home/pi/cards.dat
```

Edit with your card ids and names.

Test the program from the command line using:

```
$ sudo python reader2.py
```

Once you verify operation, edit reader2.py and turn off DEBUG messages:

```
$ nano reader2.py
```

```
DEBUG = False
```

Edit /etc/rc.local and add the following line

```
$ sudo nano /etc/rc.local
```

```
/home/pi/RFIDReaderPi/reader2.py &
```

```
exit 0
```

This will run the program any time you reboot the pi.

A log file of each door entry will be appended to:

```
/home/pi/rfiduser.log
```

```
2016-08-31 13:11:51 00000000 Start...
2016-08-31 13:11:57 E0043DBB00 NOT FOUND
2016-08-31 13:12:05 00000000 Someone Exited
2016-08-31 13:19:29 E0043DBB00 Russell Grokett
2016-08-31 13:19:37 00000000 Stop...
```

Have Fun!