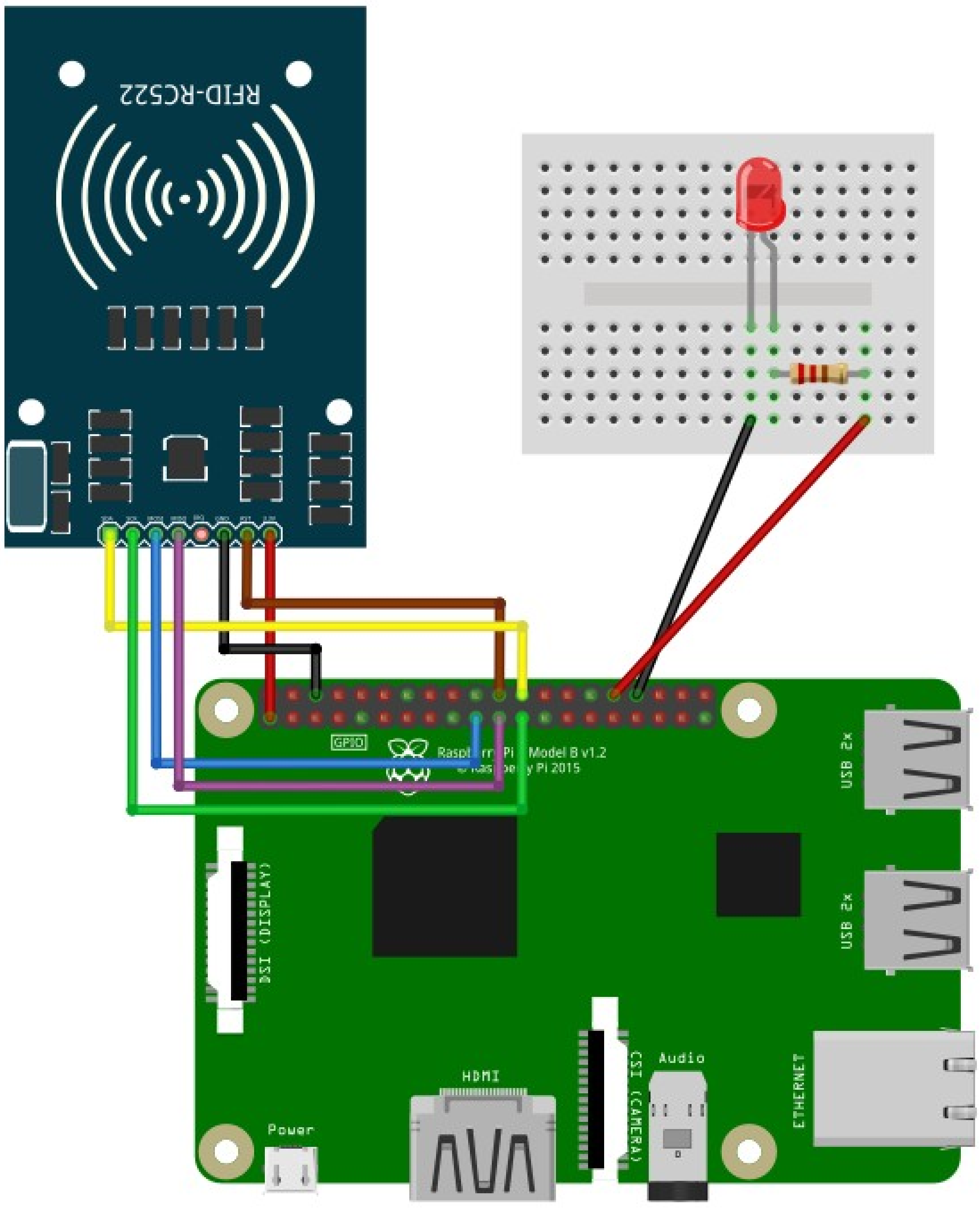
RC522 13.56MHz RFID Card Reader Project

This project shows you how to attach an RFID card reader to your Raspberry Pi, and detect when cards or keyfobs are in close proximity to the reader. Each card or fob has a unique user ID and you can make the Raspberry Pi perform an action when a valid card is detected. Our example makes a red LED light up for 5 seconds when a valid card is presented, but you could just as easily substitute the LED for a relay operating an electronic door latch.

Here’s how the Pi attaches to the RC522 reader over SPI, and to the red LED.



To enable the SPI interface on your Raspberry Pi

```
sudo raspi-config
```

Then choose option 5 Interfacing Options → P4 SPI → enable = yes → Finish. Reboot Pi.

Then install the spidev library with:

```
sudo apt-get update
sudo apt-get install python3-spidev
```

Install SPI-Py

```
cd ~
git clone https://github.com/lthiery/SPI-Py.git
cd SPI-Py
sudo python3 setup.py install
cd ~
```

Grab files from our server

```
mkdir rc522
cd rc522
wget securipi.co.uk/rc522.zip
unzip rc522.zip
ls -al
```

You should then see a list of files ending in .py

Here’s what each of them does:

MFRC522_3.py – Python3 version of main library functions for RC522 RFID reader board.

Read_3.py – Read from RFID cards. Display UID as four comma separated values.

Read_3_LED.py – Read from cards. Print card codes. Turn on LED for 5 secs if code matches.

Read_3_LED2.py – as above, but checks for two different valid cards.

Read_3_LED3.py – as above, also logs valid codes and date/time to doorlog.txt file

to run the final example

```
python3 Read_3_LED3.py
```

Change the code string in the examples to the codes for your own cards with:

```
nano Read_3_LED3.py
```

Read_3_LED.py

```
#!/usr/bin/env python
#    Copyright 2014,2018 Mario Gomez <mario.gomez@teubi.co>
#    You should have received a copy of the GNU Lesser General
Public License
#    along with MFRC522-Python.  If not, see
<http://www.gnu.org/licenses/>.

import time
import RPi.GPIO as GPIO
import MFRC522_3
import signal

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

led = 32 # GPIO.BOARD number 32 is same as GPIO.BCM GPIO12, had to
use board mode numbering, not sure why yet.
GPIO.setup(led,GPIO.OUT)

continue_reading = True
code = 0

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
    global continue_reading
    print("Ctrl+C captured, ending read.")
    continue_reading = False
    GPIO.cleanup()

# Hook the SIGINT
signal.signal(signal.SIGINT, end_read)

# Create an object of the class MFRC522
MIFAREReader = MFRC522_3.MFRC522()

# This loop keeps checking for chips. If one is near it will get
the UID and authenticate
while continue_reading:

    # Scan for cards
    (status,TagType) =
MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    # Get the UID of the card
    (status,uid) = MIFAREReader.MFRC522_Anticoll()

    # If we have the UID, continue
    if status == MIFAREReader.MI_OK:
        code = ''.join(map(str, uid))
        print("Card code =", code)
        time.sleep(1)
        if code == str(192894643156):
            print("You're In")
            # Turn on the LED on GPIO12/Pin32 for 5 seconds (This
could be a relay and electric door latch instead)
            GPIO.output(led,GPIO.HIGH)
            time.sleep(5)
            GPIO.output(led,GPIO.LOW)
```

Read_3_LED2.py

```
#!/usr/bin/env python
#    Copyright 2014,2018 Mario Gomez <mario.gomez@teubi.co>
#    You should have received a copy of the GNU Lesser General
Public License
#    along with MFRC522-Python.  If not, see
<http://www.gnu.org/licenses/>.

import time
import RPi.GPIO as GPIO
import MFRC522_3
import signal

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

led = 32 # GPIO.BOARD number 32 is same as GPIO.BCM GPIO12, had to
use board mode numbering, not sure why yet.
GPIO.setup(led,GPIO.OUT)

continue_reading = True
code = 0

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
    global continue_reading
    print("Ctrl+C captured, ending read.")
    continue_reading = False
    GPIO.cleanup()

# Hook the SIGINT
signal.signal(signal.SIGINT, end_read)

# Create an object of the class MFRC522
MIFAREReader = MFRC522_3.MFRC522()

# This loop keeps checking for chips. If one is near it will get
the UID and authenticate
while continue_reading:

    # Scan for cards
    (status,TagType) = 
MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    # Get the UID of the card
    (status,uid) = MIFAREReader.MFRC522_Anticoll()

    # If we have the UID, continue
    if status == MIFAREReader.MI_OK:
        code = ''.join(map(str, uid))
        print("Card code =", code)
        time.sleep(1)
        if code == str(192894643156):
            print("Tim You're In")
            # Turn on the LED on GPIO12/Pin32 for 5 seconds (This 
could be a relay and electric door latch instead)
            GPIO.output(led,GPIO.HIGH)
            time.sleep(5)
            GPIO.output(led,GPIO.LOW)
        if code == str(266918019540):
            print("Michelle You're In")
            GPIO.output(led,GPIO.HIGH)
            time.sleep(5)
            GPIO.output(led,GPIO.LOW)
```

Read_3_LED3.py

```
#!/usr/bin/env python
#    Copyright 2014,2018 Mario Gomez <mario.gomez@teubi.co>
#    You should have received a copy of the GNU Lesser General 
Public License
#    along with MFRC522-Python.  If not, see 
<http://www.gnu.org/licenses/>.

import datetime
import time
import RPi.GPIO as GPIO
import MFRC522_3
import signal

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

led = 32 # GPIO.BOARD number 32 is same as GPIO.BCM GPIO12, had to 
use board mode numbering, not sure why yet.
GPIO.setup(led,GPIO.OUT)

continue_reading = True
code = 0
user = "none"

def unlock():
    GPIO.output(led,GPIO.HIGH)
    time.sleep(5)
    GPIO.output(led,GPIO.LOW)

def logwrite(user):
    now = datetime.datetime.now().replace(microsecond=0)
    file = open("doorlog.txt","a")
    file.write(str(now) + '=' + str(user) + '\n')
    file.close

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
    global continue_reading
    print("Ctrl+C captured, ending read.")
    continue_reading = False
    GPIO.cleanup()

# Hook the SIGINT
signal.signal(signal.SIGINT, end_read)

# Create an object of the class MFRC522
MIFAREReader = MFRC522_3.MFRC522()

# This loop keeps checking for chips. If one is near it will get
the UID and authenticate
while continue_reading:

    # Scan for cards
    (status,TagType) =
MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    # Get the UID of the card
    (status,uid) = MIFAREReader.MFRC522_Anticoll()

    # If we have the UID, continue
    if status == MIFAREReader.MI_OK:
        code = ''.join(map(str, uid))
        print("Card code =", code)
        time.sleep(1)
        if code == str(192894643156):
            print("Tim You're In")
            # Turn on the LED on GPIO12/Pin32 for 5 seconds (This
could be a relay and electric door latch instead)
            unlock()
            user = "Tim"
            logwrite(user)
        if code == str(266918019540):
            print("Michelle You're In")
            unlock()
            user = "Michelle"
            logwrite(user)
```

This version shows the Card ID as a hex value which has a fixed length.

```
!/usr/bin/env python
#    Copyright 2014,2018 Mario Gomez <mario.gomez@teubi.co>
#    You should have received a copy of the GNU Lesser General
Public License
#    along with MFRC522-Python.  If not, see
<http://www.gnu.org/licenses/>.

import time
import RPi.GPIO as GPIO
import MFRC522_3
import signal

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

led = 32 # GPIO.BOARD number 32 is same as GPIO.BCM GPIO12, had to
use board mode numbering, not sure why yet.
GPIO.setup(led,GPIO.OUT)

continue_reading = True
code = 0

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
    global continue_reading
    print("Ctrl+C captured, ending read.")
    continue_reading = False
    GPIO.cleanup()

# Hook the SIGINT
signal.signal(signal.SIGINT, end_read)

# Create an object of the class MFRC522
MIFAREReader = MFRC522_3.MFRC522()

# This loop keeps checking for chips. If one is near it will get
the UID and authenticate
while continue_reading:

    # Scan for cards
    (status,TagType) =
MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    # Get the UID of the card
    (status,uid) = MIFAREReader.MFRC522_Anticoll()

    # If we have the UID, continue
    if status == MIFAREReader.MI_OK:
        print("Card read UID: %s,%s,%s,%s" % (uid[0], uid[1],
uid[2], uid[3]))
        a = (hex(uid[0]).split('x')[-1])
        b = (hex(uid[1]).split('x')[-1])
        c = (hex(uid[2]).split('x')[-1])
        d = (hex(uid[3]).split('x')[-1])
        e = str(a+b+c+d)
        code = e.upper()
        print(code)
        time.sleep(1)
        if code == "E673E41A":
            print("You're In")
            # Turn on the LED on GPIO12/Pin32 for 5 seconds (This
could be a relay and electric door latch instead)
            GPIO.output(led,GPIO.HIGH)
            time.sleep(5)
            GPIO.output(led,GPIO.LOW)
```