Project [RFID READER -> LCD]
User Guide

Benny Chao - A01270575
David Lee - A01251357
November 6th, 2022

Purpose

RFID to LCD supports:

- reading data from using a RFID (radio-frequency identification) reader
- writing data to an RFID card using a RFID reader
- sending data read from a client (RFID side) to the server (LCD side)
- displaying the data received from the client on a LCD (liquid-crystal display)

Installing

Obtaining

```
git clone https://github.com/Bnyaoo/rudp_rfid_to_lcd
```

Building

Server:

```
cd server
mkdir cmake-build-debug
cmake -S .-B cmake-build-debug
cmake --build cmake-build-debug
```

Client:

```
cd client
mkdir cmake-build-debug
cmake -S .-B cmake-build-debug
cmake --build cmake-build-debug
```

The compiler can be specified by passing one of the following to cmake:

- `-DCMAKE_C_COMPILER="gcc" -DCMAKE_CXX_COMPILER="g++"`
- `-DCMAKE_C_COMPILER="clang" -DCMAKE_CXX_COMPILER="clang++"`

Running

Server:

```
./server -p <port number>
```

Client:

```
./client -o <server address> -p <port number>
```

Hardware Setup

Obtaining

purchase
SunFounder Raspberry Pi Davinci Starter Kit
Raspberry Pi 400 Personal Computer Kit

Setup

Server:

Components:

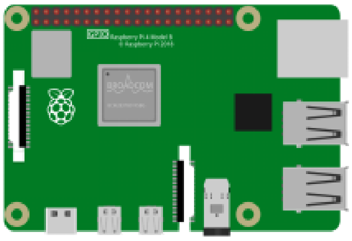
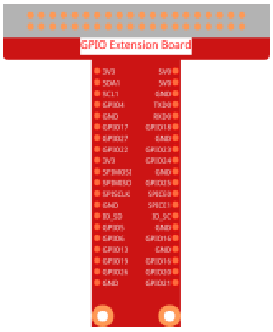
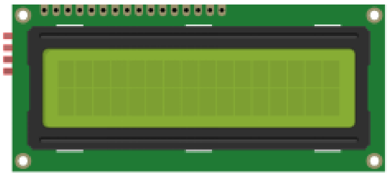
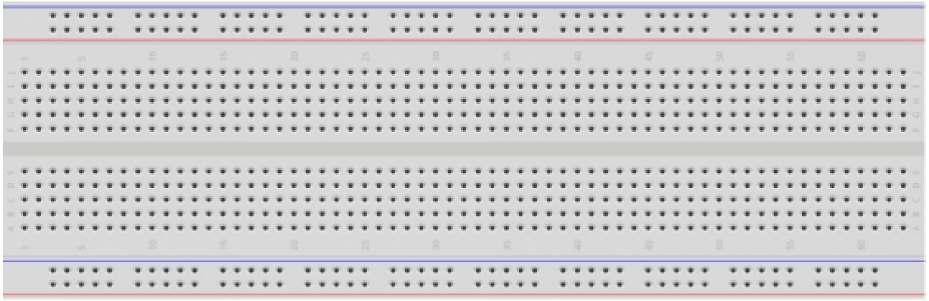
<table>
<tr><td>1 * Raspberry Pi
</td><td>1 * T-Extension Board
 </td><td>1 * I2C LCD1602
</td></tr>
<tr><td colspan="2">1 * 40-pin Cable
 </td><td>4 * Jumper Wires F/M
 </td></tr>
<tr><td colspan="3">1 * Breadboard
</td></tr>
</table>

Figure 1. Electronic components required for server side

Schematics:

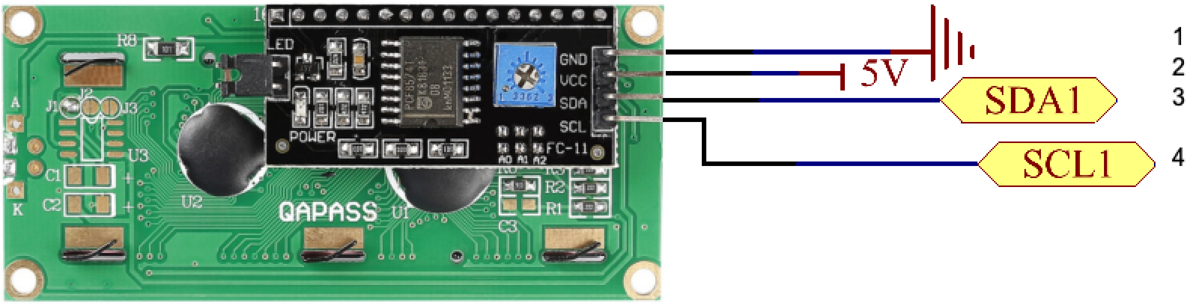
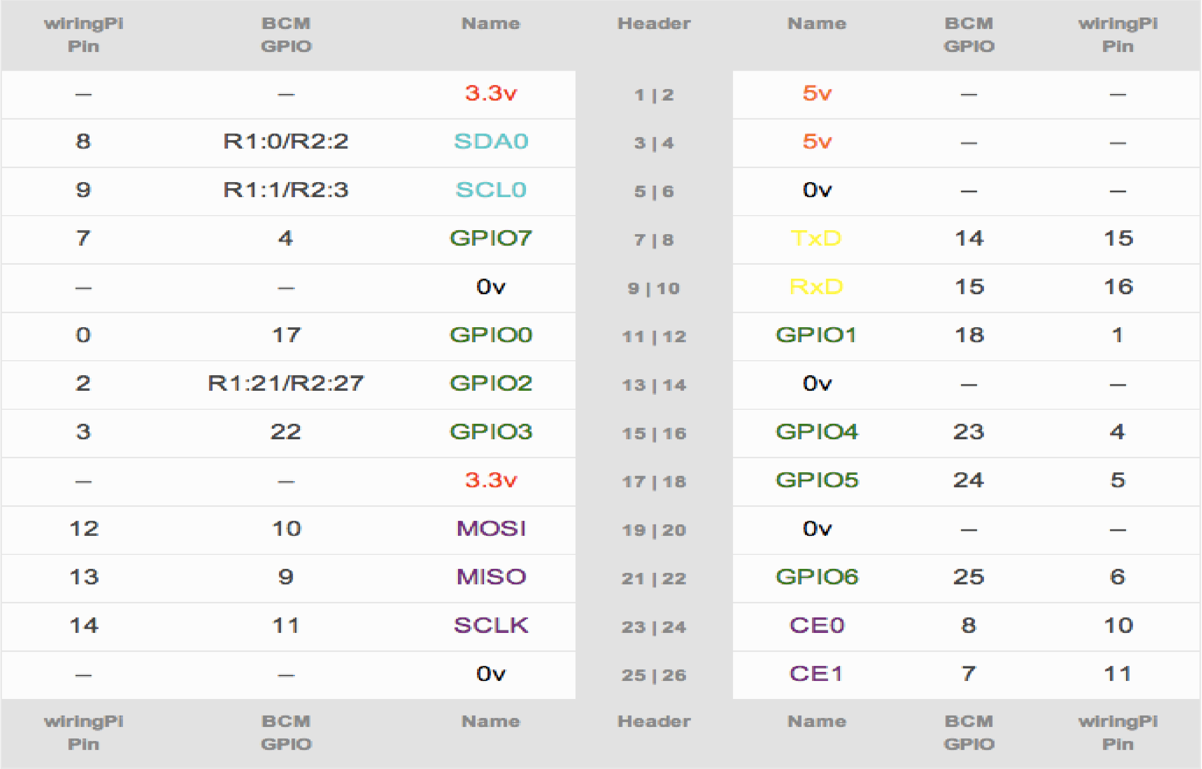


Figure 2. Schematic diagram of LCD labelled with numbers and T-board names

T-board Name	Physical (header)
GND (1)	pin 6
VCC (2)	pin 4
SDA (3)	pin 3
SCL (4)	pin 5

Figure 3. T-board name to physical pin table for LCD module

The physical pins are defined by the wiringPi board numbering system illustrated below.



wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin
–	–	3.3v	1 | 2	5v	–	–
8	R1:0/R2:2	SDA0	3 | 4	5v	–	–
9	R1:1/R2:3	SCL0	5 | 6	0v	–	–
7	4	GPIO7	7 | 8	TxD	14	15
–	–	0v	9 | 10	RxD	15	16
0	17	GPIO0	11 | 12	GPIO1	18	1
2	R1:21/R2:27	GPIO2	13 | 14	0v	–	–
3	22	GPIO3	15 | 16	GPIO4	23	4
–	–	3.3v	17 | 18	GPIO5	24	5
12	10	MOSI	19 | 20	0v	–	–
13	9	MISO	21 | 22	GPIO6	25	6
14	11	SCLK	23 | 24	CE0	8	10
–	–	0v	25 | 26	CE1	7	11
wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin

Figure 4. wiringPi board numbering system diagram

After the circuit has been built, the server-side setup should resemble the diagram below.

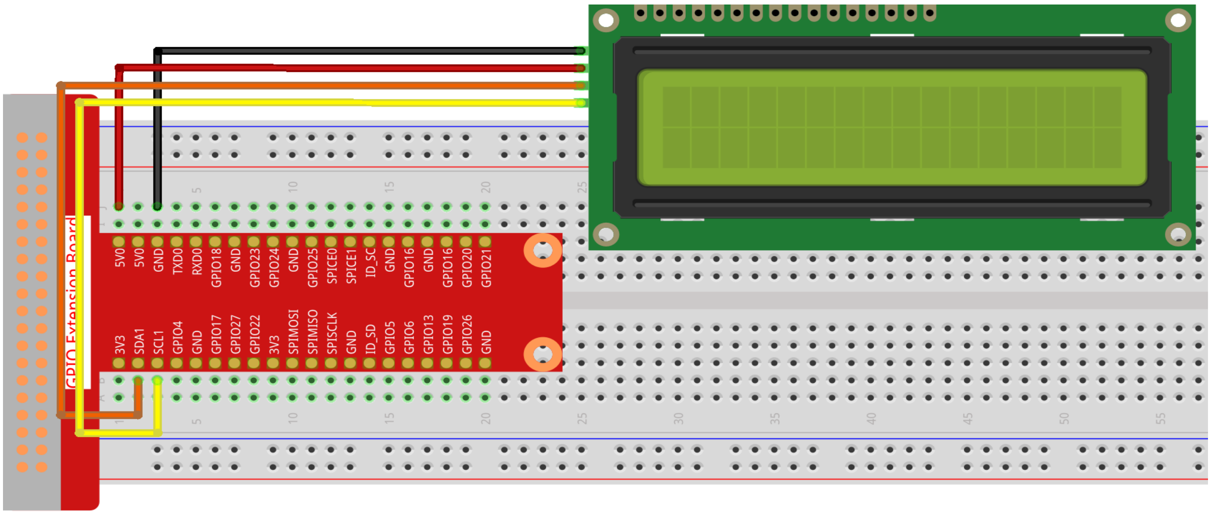


Figure 5. Diagram of LCD hooked up to breadboard connected to pi with a GPIO extension

If I2C hasn't been configured on your Raspberry Pi, follow this link to setup your Pi to detect the LCD.

Client:

Components:

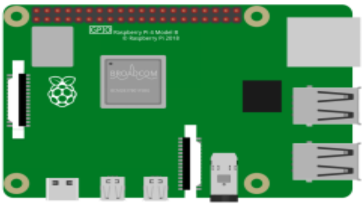
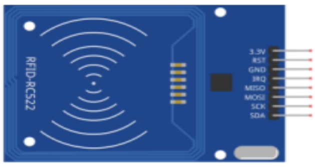
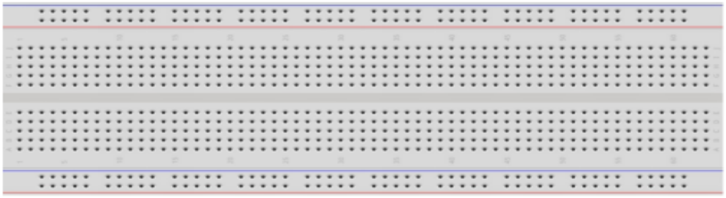
1 * Raspberry Pi 	1 * T-Extension Board	1 * RFID RC522 (with white card and key tag)  RFID-RC522 3.3V RST GND IRQ MISO MOSI SCK SDA
		9 * Jumper Wires M/M
1 * 40-pin Cable		
1 * Breadboard 		

Figure 6. Electronic components required for client side

Schematics:

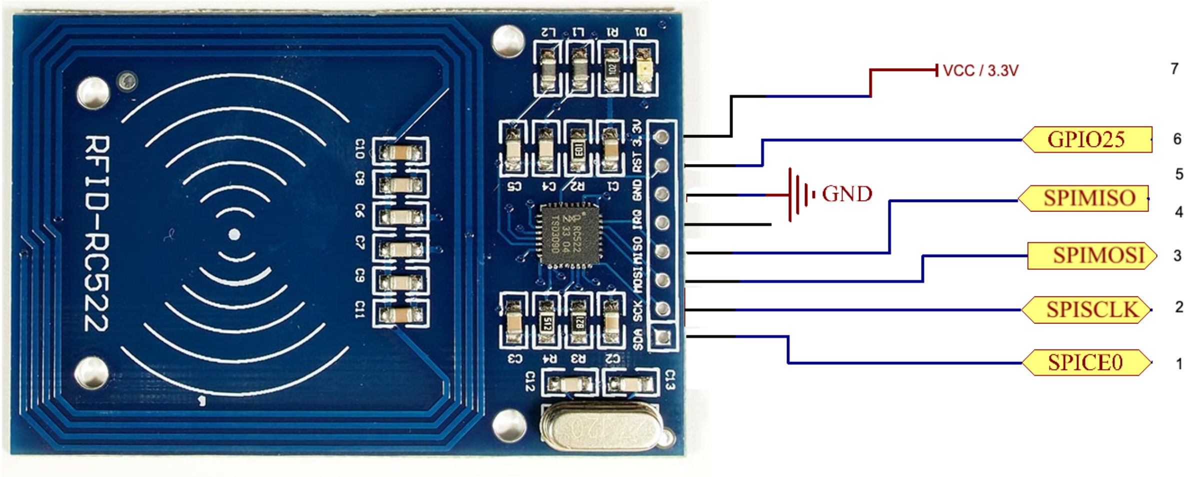


Figure 7. Schematic diagram of MFRC522 RFID labelled with numbers and T-board names

T-board Name	Physical (header)
SDA (1)	pin 24
SCK (2)	pin 23
MOSI (3)	pin 19
MISO (4)	pin 21
IRQ	no connection
GND (5)	pin 25
RST (6)	pin 22
VCC (7)	pin 17

Figure 8. T-board name to physical pin table for RFID module

NOTE: IRQ is an interrupt pin that alerts the microcontroller when an RFID tag is in the vicinity and is intended to be left unconnected because the library used in this project does not support it.

If reference is required for setting up your project, refer to Figure. 4 for a visual representation of the wiringPi board numbering system.

After the circuit has been built, the client-side setup should resemble the diagram below.

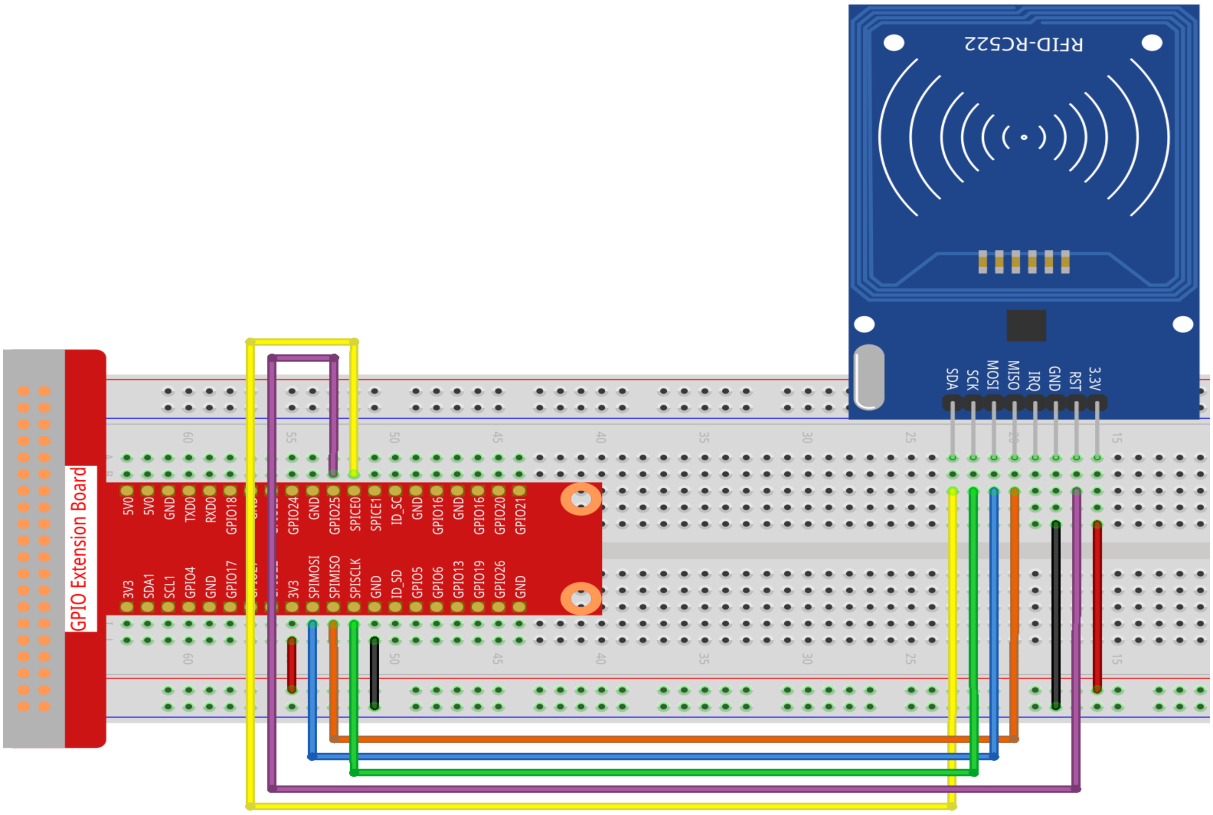


Figure 9. Diagram of RFID hooked up to breadboard connected to pi with a GPIO extension

If SPI hasn't been configured on your Raspberry Pi, follow this link to setup your Pi to detect the RFID module.

Environment Variables

The following environment variables alter the behaviour of rfid_to_lcd

Variable	Purpose
./build/server	Initialises all variables and starts the UDP server listening to the default port.
./build/server -p <port no.>	Initialises all variables and starts the UDP server, listening to the specified port.
./build/client -o -p	Initialises all variables and starts the UDP client connected to specified server address and port.

Features

- rfid_to_lcd supports executing UDP data transfer using network sockets and electronic components.
- The server receives UDP packets from the client and displays the data on a LCD connected via a T-board extension to the RaspberryPi.
- The client sends UDP packets to the server via data written in an RFID card.
    - The client is prompted to write to stdin and the program waits until an RFID card is tapped against the rc522 RFID module to store the data to the card.
- The client waits to receive an ack packet from the server before another data packet can be sent.

Built-in Commands

rfid_to_lcd supports the following built-in commands:

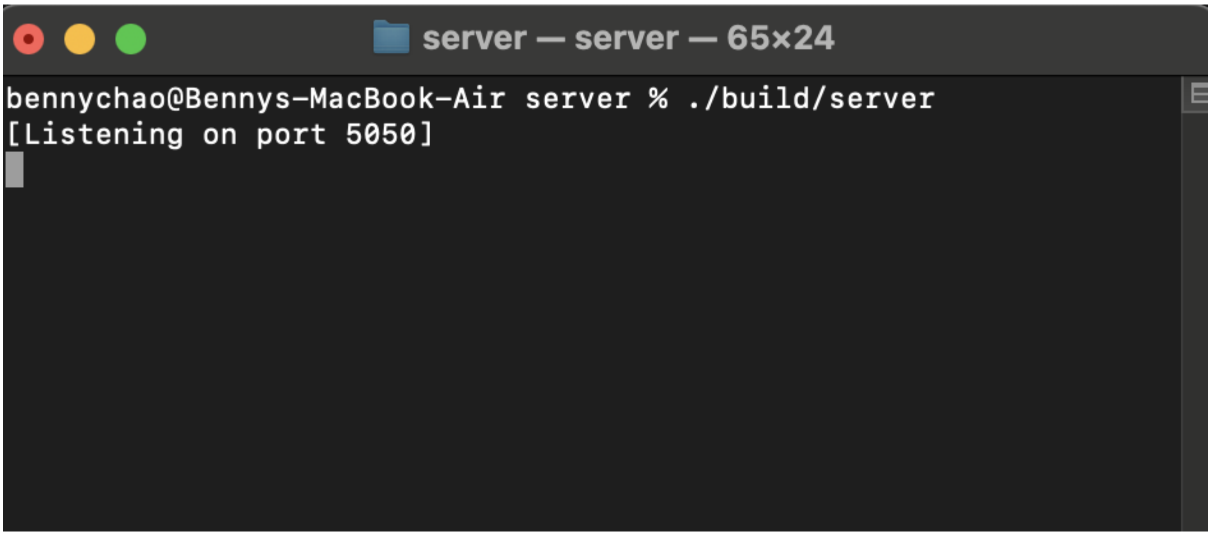
Command	Purpose
stdin	Prompts client to write to a char buffer and stores the data into an rfid card which is sent to the server.

Limitations

- Initialization for the client must adhere to the following format
    - Usage: ./build/client -o <proxy server address> [-p port_number]
- Port specified must be available
- Data is limited to a maximum of 16 characters for the client to match the number of cells available in the LCD on the server-side.

Examples

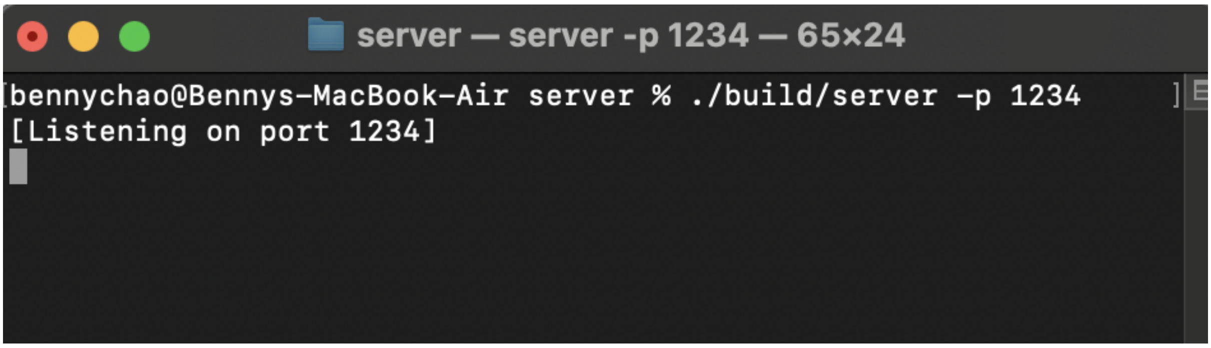
Running the server



```
bennychao@Bennys-MacBook-Air server % ./build/server
[Listening on port 5050]
```

Figure 10. Initialising the server program on the default port

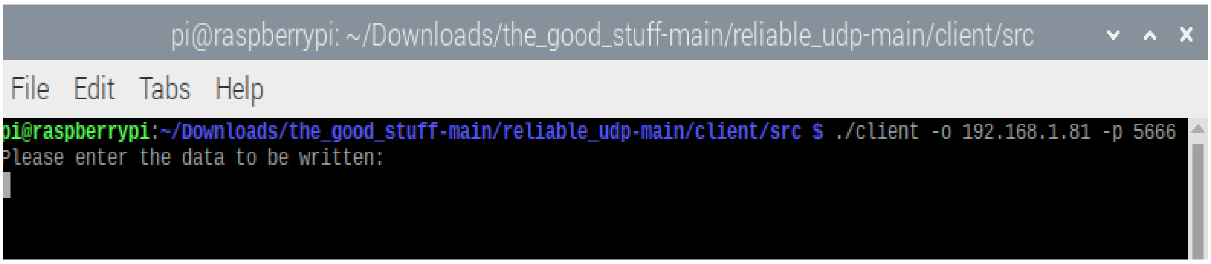
Running the server w/ a specific port



```
[bennychao@Bennys-MacBook-Air server % ./build/server -p 1234      ]
[Listening on port 1234]
```

Figure 11. Initialising the server program on a specified port

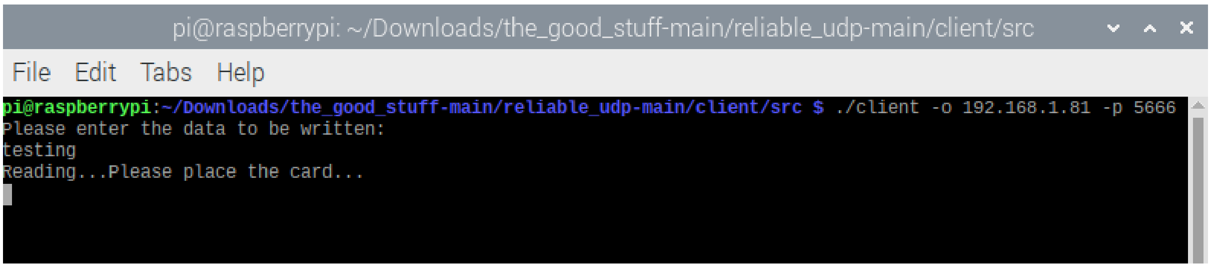
Running the client


pi@raspberrypi: ~/Downloads/the_good_stuff-main/reliable_udp-main/client/src
File
Edit
Tabs
Help

```
pi@raspberrypi:~/Downloads/the_good_stuff-main/reliable_udp-main/client/src $ ./client -o 192.168.1.81 -p 5666
Please enter the data to be written:
```

Figure 12. Initialising the client program on a specified host address and port number

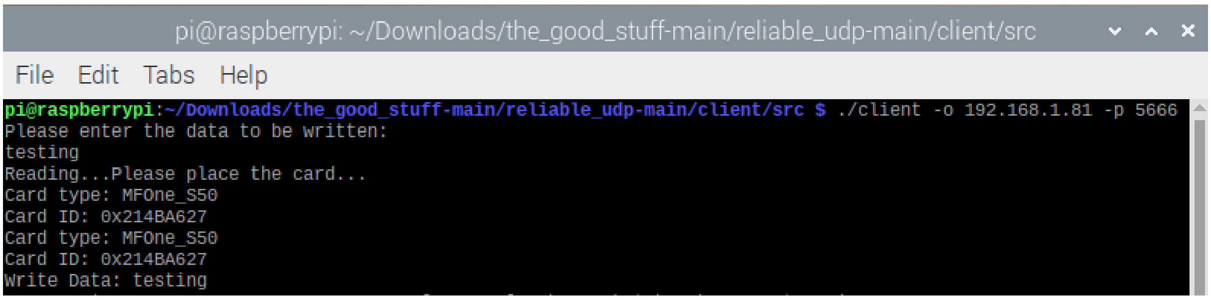
Writing to stdin and waiting for RFID card tap Client-Side


pi@raspberrypi: ~/Downloads/the_good_stuff-main/reliable_udp-main/client/src
File Edit Tabs Help

```
pi@raspberrypi:~/Downloads/the_good_stuff-main/reliable_udp-main/client/src $ ./client -o 192.168.1.81 -p 5666
Please enter the data to be written:
testing
Reading...Please place the card...
```

Figure 13. String is provided by the user and the program waits for a card tap

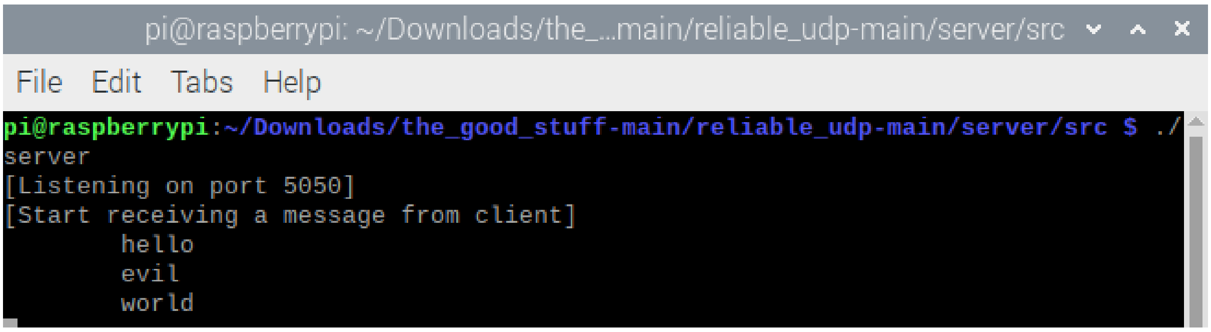
Writing data to RFID card


pi@raspberrypi: ~/Downloads/the_good_stuff-main/reliable_udp-main/client/src
File Edit Tabs Help

```
pi@raspberrypi:~/Downloads/the_good_stuff-main/reliable_udp-main/client/src $ ./client -o 192.168.1.81 -p 5666
Please enter the data to be written:
testing
Reading...Please place the card...
Card type: MFOne_S50
Card ID: 0x214BA627
Card type: MFOne_S50
Card ID: 0x214BA627
Write Data: testing
```

Figure 14. rc522 detects a RFID card; reading from stdin and writing to the card

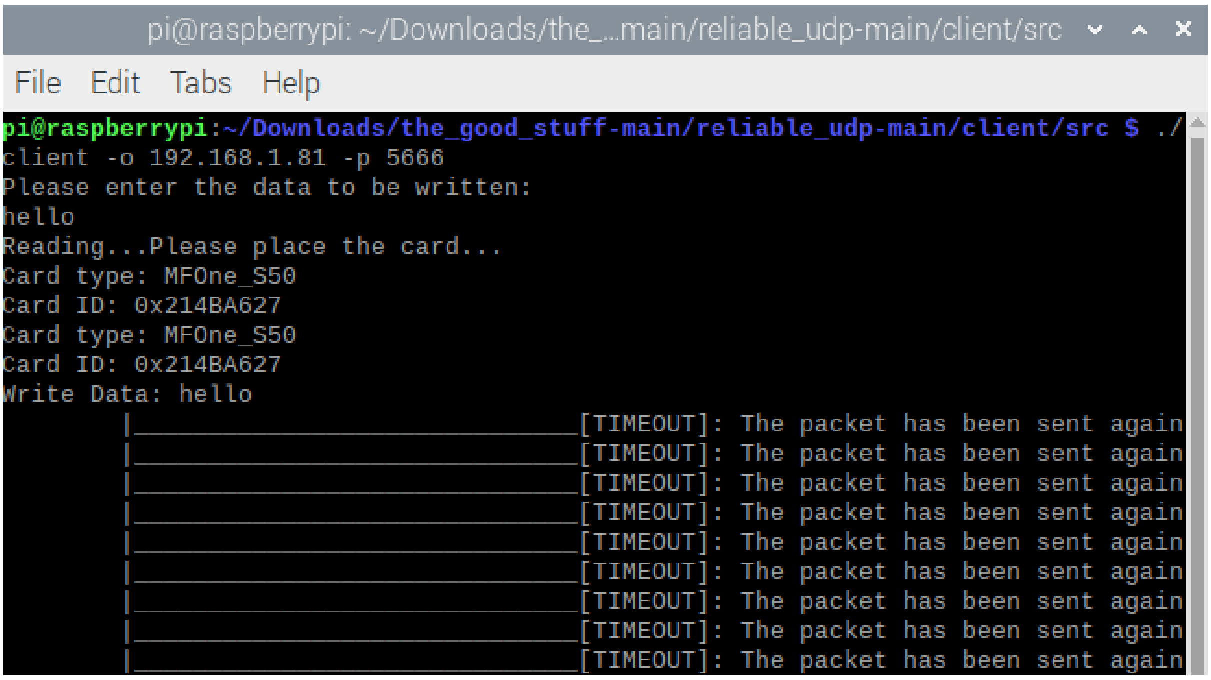
Receiving data from RFID on Server-Side


pi@raspberrypi: ~/Downloads/the_...main/reliable_udp-main/server/src
File Edit Tabs Help

```
pi@raspberrypi:~/Downloads/the_good_stuff-main/reliable_udp-main/server/src $ ./
server
[Listening on port 5050]
[Start receiving a message from client]
        hello
        evil
        world
```

Figure 15. A sequence of data from rfid card taps, displayed in server program

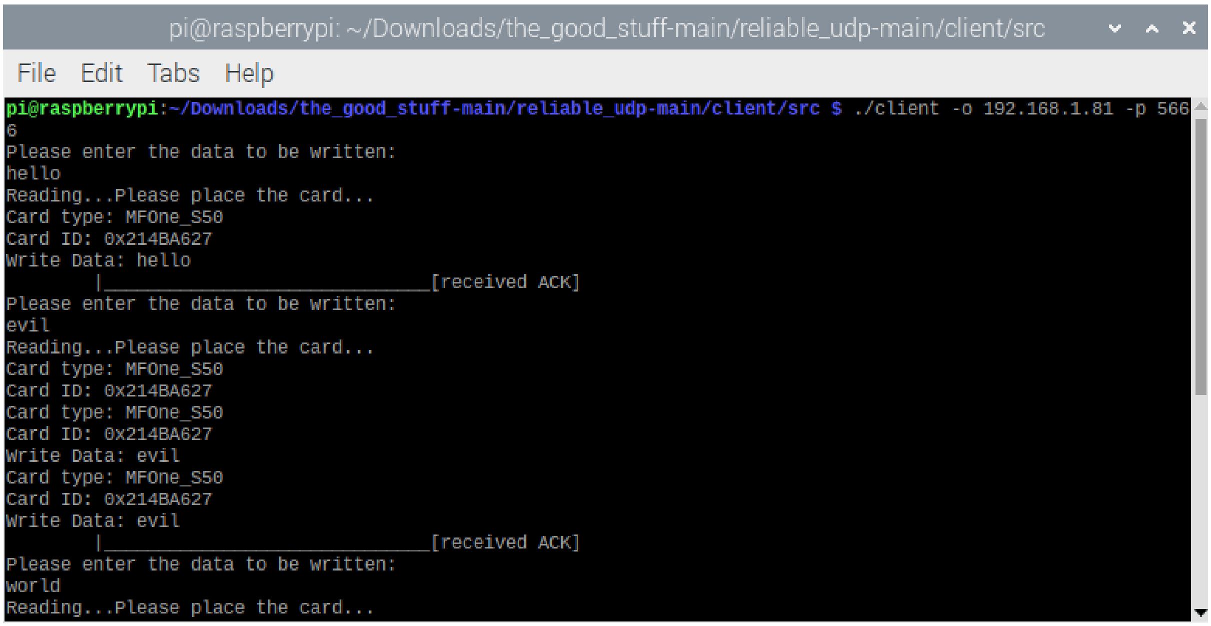
Waiting for ACK on Client-Side


pi@raspberrypi: ~/Downloads/the_...main/reliable_udp-main/client/src
File Edit Tabs Help

```
pi@raspberrypi:~/Downloads/the_good_stuff-main/reliable_udp-main/client/src $ ./
client -o 192.168.1.81 -p 5666
Please enter the data to be written:
hello
Reading...Please place the card...
Card type: MFOne_S50
Card ID: 0x214BA627
Card type: MFOne_S50
Card ID: 0x214BA627
Write Data: hello
        |______________________________[TIMEOUT]: The packet has been sent again
        |______________________________[TIMEOUT]: The packet has been sent again
        |______________________________[TIMEOUT]: The packet has been sent again
        |______________________________[TIMEOUT]: The packet has been sent again
        |______________________________[TIMEOUT]: The packet has been sent again
        |______________________________[TIMEOUT]: The packet has been sent again
        |______________________________[TIMEOUT]: The packet has been sent again
        |______________________________[TIMEOUT]: The packet has been sent again
        |______________________________[TIMEOUT]: The packet has been sent again
```

Figure 16. Packet unsuccessfully sent, results in retransmission in client program

Receiving an ACK


pi@raspberrypi: ~/Downloads/the_good_stuff-main/reliable_udp-main/client/src
File Edit Tabs Help

```
pi@raspberrypi:~/Downloads/the_good_stuff-main/reliable_udp-main/client/src $ ./client -o 192.168.1.81 -p 566
6
Please enter the data to be written:
hello
Reading...Please place the card...
Card type: MFOne_S50
Card ID: 0x214BA627
Write Data: hello
        |______________________________[received ACK]
Please enter the data to be written:
evil
Reading...Please place the card...
Card type: MFOne_S50
Card ID: 0x214BA627
Card type: MFOne_S50
Card ID: 0x214BA627
Write Data: evil
Card type: MFOne_S50
Card ID: 0x214BA627
Write Data: evil
        |______________________________[received ACK]
Please enter the data to be written:
world
Reading...Please place the card...
```

Figure 17. Successful data transfer between client and server