An investigation into the creation of an RFID Door Lock – Artefact Report

Mathew Mark Allington

St Brendan's Sixth Form

Identification Details

Candidate Number: 0031

Centre Number: 50739

Abstract

The device I investigated was an RFID door lock. RFID stands for Radio Frequency Identification Device and it has the ability to read and write key cards with RFID compatible chips. This is then used to grant access to open a door. The door lock comprised of 4 main elements. The first element is the door lock itself; this will be a magnetized door rim release where an electro magnet stops the door from opening at the frame. The second element is the storage, addition and deletion of user's RFID door tags. The third is control panel that would include an LCD crystal display that the user on the inside of the door can use to view the status and toggle on/off of the locking mechanism. The fourth is the RFID reader itself, which would read the UID so that the Arduino can check it against the users in its database.

Keywords: RFID, Door Lock, Arduino

Table of Contents

An investigation into the creation of an RFID Door Lock – Artefact Report

Radio Frequency identification (RFID) technology is utilized in many aspects of every day life including controlling the access over entire campuses, or even on the shop floor keeping track of stock or even preventing theft [1]. We trust this technology to secure our assets and buildings without a second thought into the workings of such tech. I decided investigate one implementation of the technology, an RFID door lock. In order to have a deep understanding of the key mechanisms required to create a fully functioning device. I investigated the exact protocols these types of systems use to verify and exchange card information and then explored the individual components involved in manufacturing such a device. Will then complement the implementation of these concepts into a working artefact.

1. Requirements and Specifications

My first step in this process was to complete some preliminary analysis of the core features that underpin these types of systems. Here I have constructed a table of my findings (refer to Table 1):

Table 1: Core Functions

Function	Requirement
User Interface	To successfully create a system in which I can add remove users – this will enable me to add and remove users via a master card and an on screen UI on the LCD crystal display. It would also enable me to potentially black list cards and maybe set temporary ones.
Card Storage	Store UID numbers on an SD module or internal Arduino storage, this storage device will allow me to create a database of card
Card recognition	UIDs are able to lock and unlock the door. Get the RFID module to read UIDs – this will allow me to keep track of the different cards that are trying to access the door.
Door Locking Component	Get the magnetic door lock to open and close – this will mean having to source and electric door lock and creating a switching system that allows the Arduino to operate a higher current that would be passed through the coil of the lock.

These core functions will form the basis of my research and will help guide the necessary research required for the project.

2. Components and Communication Protocol Analysis

2.1 Architecture

For my project I chose the Arduino architecture to base my project upon. Arduino provides a high level programming interface that allows C code to be converted into raw assembler and flashed onto the chip [2]. This provided me with extra flexibility and ease when implementing complex protocols such as the RFID protocol used to extract the data from the card.

2.2 Components Required

So naturally, all of the modules had to be compatible with both the voltage requirements and the kind of communication protocols used by the Arduino microcontroller. I carefully selected my modules based on the overall functionality they would contribute to the final device and the key objectives that they hit (See Table 2).

Table 2: Components Required

Module Name	Description	Objectives achieved by module
SD Module	A device used to read a memory card and input the data into a main controller. [3]	Card Storage
LCD Crystal Display	A 16x2 character display capable of displaying up to 32 characters and even custom ones. [4]	User Interface
RFID Card Reader / Writer	A device capable of reading wireless / RFID based cards. This particular module can also re-writing them. [5]	Card Recognition
Arduino Micro-Controller	A device capable of controlling pre-programmed digital inputs and outputs. [6]	Simultaneous use of all objectives
Electro Magnetic Door Lock	A magnet based system which locks and unlocks a door at the request of the main micro controller.	Door Locking Component

2.3 Component Networking

In order to use these components, I needed to find out how they function and how I am able to send and receive data from them. I found that the more data driven components such as the RFID card reader and the SD module heavily relied on a communication protocol called the Serial Peripheral Interface (SPI). This protocol allows data to be send around a controller area network (CAN) [7] where multiple devices can communicate with each other using encoded packets. Figure 1 shows the network diagram for my device, as you can see, one module uses an IC2 backpack to minimize is pin usage by swapping the use of analog signals for digital ones.

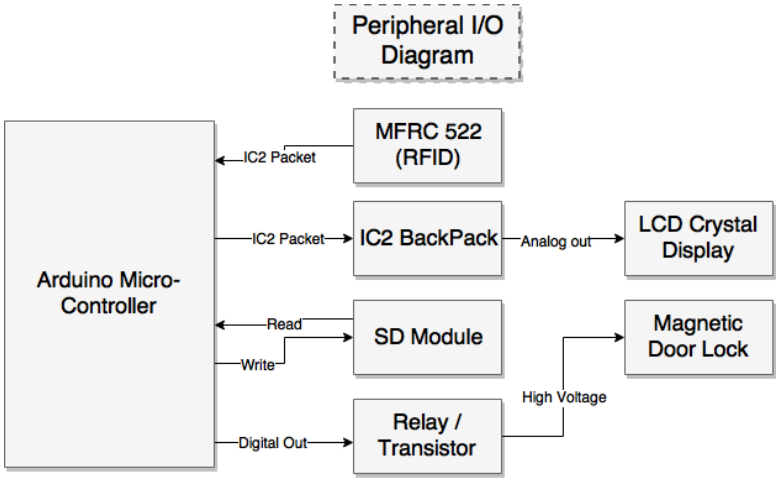


Figure 1

2.4 Serial Peripheral Interface

Understanding the SPI protocol was essential to the success of my project, I could not only use this protocol to create a network where all communication between devices could be properly governed, I could also diagnose its faults. There are 4 main pins to be concerned with when setting up an SPI based module (SCK, MISO, MOSI, CS), the other two are power and ground used to power the module (VCC, Ground). The SCK, MOSI and MISO pins are used to communicate with the board via synchronous serial communication, this means that two devices communicate by sending data packets from the transmitting side to the receiving side according to a synchronised clock. The synchronised clock allows the hardware

to communicate concurrently without overlap [8]. This is important because serial devices only communicate using one line, as opposed to parallel lines which utilise multiple lines (Figure 2).

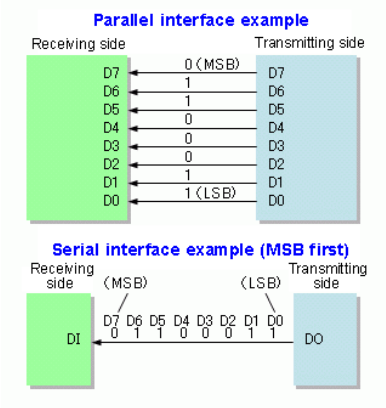


Figure 2 [9]

In this particular context, the Arduino acts as a Master and the SPI module acts as a slave device. This means that Arduino governs the SPI module. For example, the SD module (peripheral / slave) acts as an input device for the data being read from the SD module. Data is passed in (Slave -> Master) using the MISO serial pin, and then passed back out (Master -> Slave) using the MOSI serial pin, a typical serial transmission may look like this (Figure 3).

Start Bit	1	0	0	0	0	1	1	0	Stop Bit

Figure 3

Using the start and stop bit, the receiving side detects when the data packet stops. The synchronization of the whole packet is coordinated by the SCK pin (Serial Clock). This is hosted by the master and is useful when using a system with multiple peripherals, similar to my RFID Lock, and my project also includes the RFID module which utilises the same SPI interface. The CS pin is not proprietary to the SPI interface, this is a separate pin that is used for initialising and resetting the SD module [3].

2.5 Radio Frequency Identification

The type of chip I will be using is passive, Passive tags do not contain their own independent power supply and rely on the inductive coil in the reader to supply its power. When the Chip is placed in close proximity to the reader, the chip is powered up and it transmits the contents of its sectors back to the

reader. The chip I was using contained 60 sectors, where the numbering system started at zero. The normal convention states that the UID is stored in sector 0, in order to maintain conformity with any card this will be the same for my reader. The UID is consists of 8 Hexadecimal digits which can be used to validate the card.

3. Module Implementation

For each module the wiring varies based on the module function. However, a common themed arrived which showed that almost all of the modules utilised the SPI protocol found in my preliminary research.

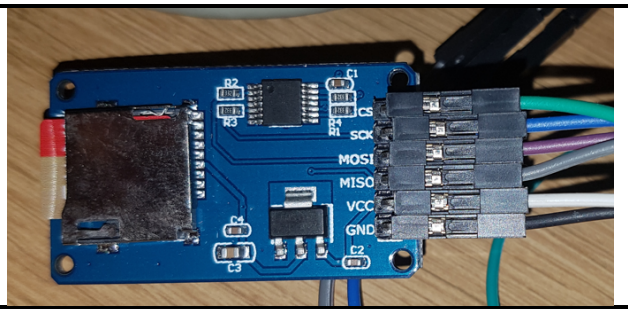
3.1 SD Module

The data flow of the SD module is bi-directional due to the requirements to read and write the data onto the card, hence requiring both MISO (Master in slave out) and MOSI (Master Out Slave In) pins [3]. Another thing to note about this particular module, the File System was limited to cards with capacities 32GB or more. If the SD was smaller, then the SD card would fail the initialisation stage of the setup.

Wiring Configuration

Arduino to SD Module wiring (Table 3)

Arduino Uno Pins	SD Module Pins
5V	5V
Ground	Ground
13	SCK
12	MISO
11	MOSI
4	CS



Main commands

To Store and read the UIDs from the SD card, I utilized the commands from the Arduino SD library :

Syntax:

```
#include <SPI.h> //Imports a library containing the necessary sub-routines for SPI communication.
#include <SD.h> //Imports a “File” object and tools necessary for initialising the SD module.
```

```
Serial.begin(x); //Would initialise the SD card using pin x, this is the CS pin
File fileObject = SD.open("File Location", MODE); // This line creates a new file object, this stores the mode (either Write or Read) and the SD location of the file being addressed.
fileObject.println("Line of text"); //using the file object, it appends to the text file
String inp = fileObject.read(); //This reads a particular part of a stream from an SD file that is being read.
fileObject.close(); //This safely cleans up any open streams associated with the SD file, this decreases the chance of file corruption.
```

Post Processing

I utilised these commands to capture a CSV (comma separated values) file which I stored the UID of each card:

UIDS.CSV:

FFFFFFFF,

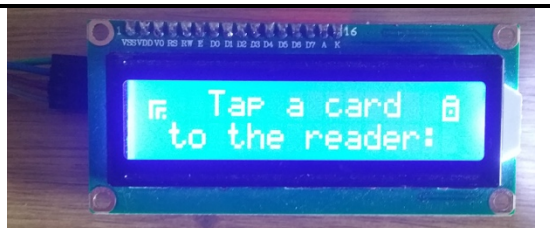
A7D4FAC7,

3.2 LCD Crystal Display

Wiring Configuration

Arduino to LCD Display wiring (Table 4)

Arduino Uno Pins	IC2 Back Pack
5V	5V
Ground	Ground
Analogue 4	SDA
Analogue 5	SCL



IC2 Backpack (Digital) to LCD Crystal display (Analogue)

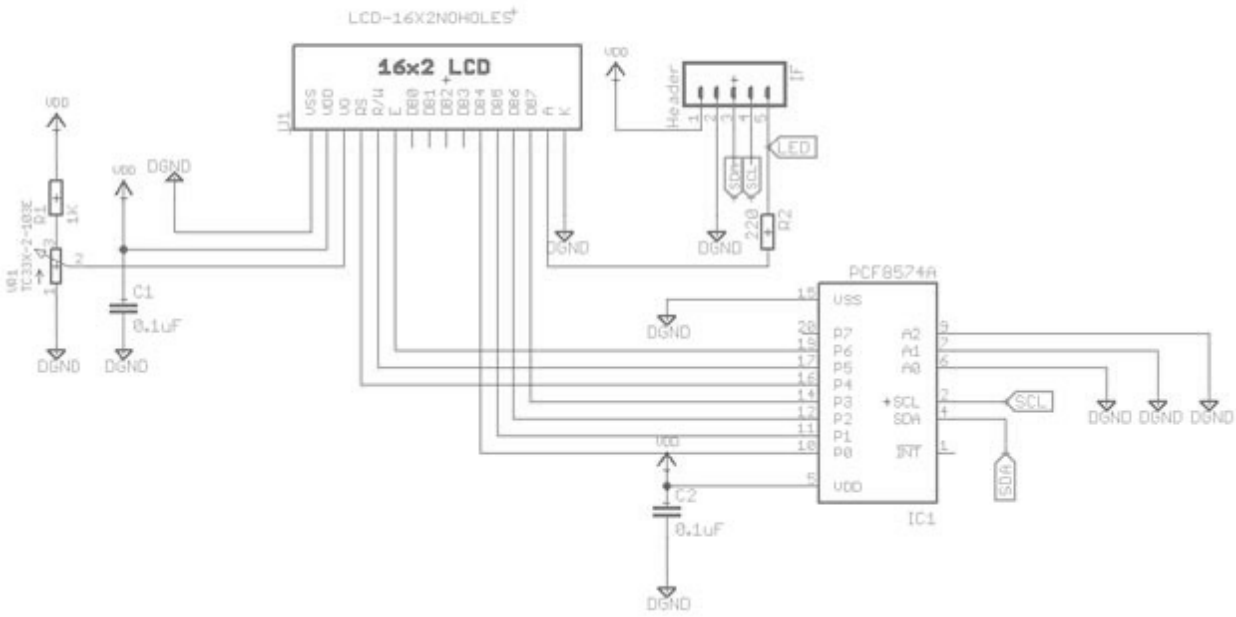


Figure 4 [10]

As you can infer from figure 4, the IC2 backpack utilises the same SPI protocol using the same pins (denoted by SDA and SCL). To use the module I simply soldered the relevant 16x2 LCD pins to the solder points that connect to the circuit board that holds the PCF8574A chip [10]. See Figure 5 reference:

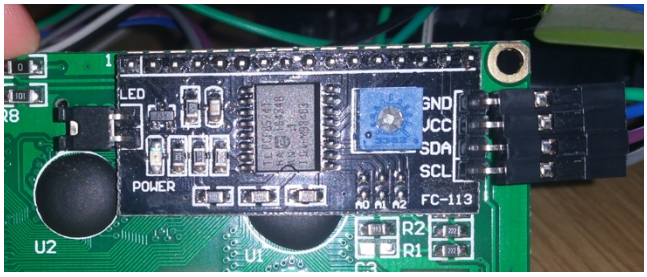


Figure 5

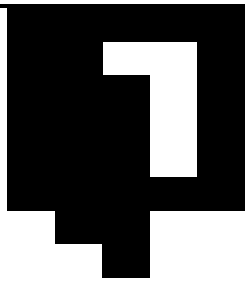
Icon Creation:

My project plan proposed that the crystal could have custom characters would act as different icons for the different menus of the display. I found that to create an icon, you have to hard code it into a byte array, which signifies the on/off of each pixel. So for example, take to open door menu (figure 6). The byte

array for this particular icon would look like table 5. I found a useful online editor which allowed me to draw the individual pixels and then output it into a byte array format [11].

Table 5 (Icon creation)

```
byte openChar[8]
= {
  0b11111,
  0b11001,
  0b11101,
  0b11101,
  0b11101,
0b11111,
  0b01100,
  0b00100
}; //byte array
```



I then created methods that would display a particular menu and custom text, so for example the removing card menu looked like this figure 7

Figure 6

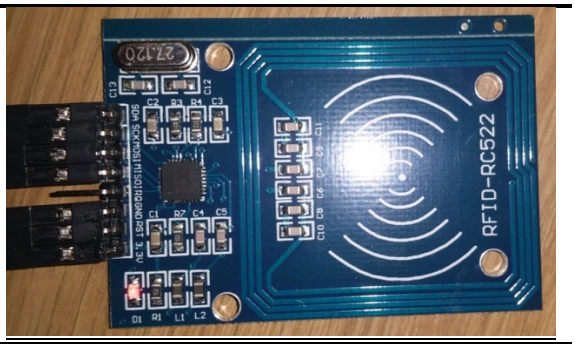
Figure 7

3.3 RFID Module

Wiring

Arduino to RFID wiring (Table 6)

Arduino Uno Pins	RFID Module
3.3V	3.3V
Ground	Ground
13	SCK
12	MISO
11	MOSI
10	SDA
9	RST (Reset)



Main Commands

To control the MFRC522 I used an open source library from Git Hub [12]. The library allows me to perform the three basic functions required for my door lock:

1. Detect if a card is present
2. Detect the particular type of card
3. Read the NUID

Syntax:

```
SPI.begin(); // Initialises SPI bus

rfid.PCD_Init(); // Initialises MFRC522 (RFID Module)

key.keyByte[i] = 0xFF; // Sets the authentication key, for no authentication
the key is just 0xFF (Blank)

rfid.PICC_IsNewCardPresent(); //This checks to see if card is present
(Returns true or false)

rfid.PICC_ReadCardSerial(); //Verifies whether the card has been read

MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak); // Obtains the
type of card, Types: MFRC522::PICC_TYPE_MIFARE_MINI,
MFRC522::PICC_TYPE_MIFARE_1K, MFRC522::PICC_TYPE_MIFARE_4K
```

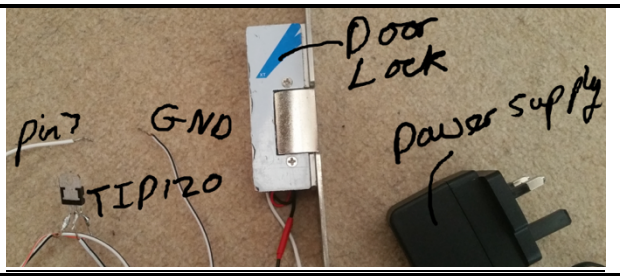
```
currentCard[i] = rfid.uid.uidByte[i]; // Where i = index number for the cards memory, it reads that specific byte in the card memory
```

3.4 Magnetic Door Lock

Wiring

Arduino to Door Lock (Table 7)

Arduino Uno Pins	TIP 120
Pin 7	Base
Ground	Collector / Common Ground



Findings

Due to the Arduino microcontroller can only output a maximum of 5v [6]and the door lock required a PD (potential difference) of 12v and a current of 1 amp, I needed a means of using the Arduino micro controller to switch on/off this higher voltage supply. Due to knowledge in previous endeavours of mine, I knew that I could use A TIP 120 transistor (see Figure 8) to pass in a voltage of 5v, then switch on the ground to the door lock, which means current will flow round, enabling the electro magnet (see Table 7) to switch on. This then creates a solid lock on the door, where the door is then unable to open back up again.

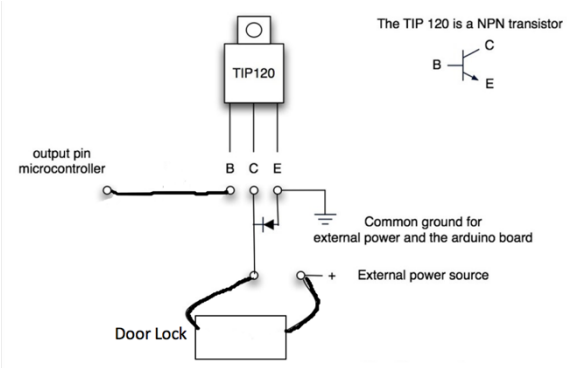


Figure 8

4. Conclusions

My main conclusion is that the SPI protocol was an integral mechanism for the final product. When creating a device with such complexity and a wide range of different modules, there has to be a common method for them to interact. This interface allows multiple devices to communicate with a central master controller in a concurrent manner. This allows an infinite number of devices on the CAN (Controller Area Network) [7] within the limitations of the hardware driving it. Although, I have found that this method is very resource intensive, and half way through the project I was forced to upgrade the micro-controller from an Arduino UNO (32kb) to and Arduino Mega 2560 (256kb flash) due to the UNO having insufficient flash memory to load the complete firmware on to [13].

With regards to the reproducibility of my final product, I have concluded that my project can in fact be reproduced by any one with the use of sufficient documentation. The availability of parts available online and the wide range of high level programming support makes the project a viable option for any DIY enthusiast like myself to build their own home door lock system. This success was due to the Arduino platform that compiles C/C++ to raw assembly and flashing it directly to the micro-controller making it far easier to create more complex programs more efficiently.

My findings also reveal that the research into the specific protocols and mechanisms of each component has played a pivotal role in the success of the project. Have knowledge basic fundamental knowledge on the SPI protocol has assisted with solving problems I have encountered, such as a problem I was having with the SD initialization. In this particular situation, I was unable to initialize the SD card properly, with my knowledge of the SPI protocol (refer to 2.4) I found out that there was a problem with the CS pin which is responsible for initializing the SPI bus. Because they were sharing the same pin there was a clash of data packets. To fix this I have override he pin it operates on be using the #define feature in C++.

Bibliography

[1] W. D. D.Hahnel, "Mapping and localization with RFID technology," Freiburg University.

[2] Arduino LLC, "What is arduino?," 2015. [Online]. Available: http://web.csulb.edu/~hill/ee400d/Technical%20Training%20Series/02%20Intro%20to%20Arduino.pdf.

[3] Arduino LLC, "SD Library Documentation," [Online]. Available: https://www.arduino.cc/en/Reference/SDbegin.

[4] Arduino LLC, "LCD Crystal library," [Online]. Available: https://www.arduino.cc/en/Reference/LiquidCrystal.

[5] Arduino LLC, "MFRC522," 2013. [Online]. Available: https://playground.arduino.cc/Learning/MFRC522.

[6] Arduino LLC, "Arduino Mega 2560," Arduino, [Online]. Available: https://www.arduino.cc/en/Main/arduinoBoardMega2560/.

[7] K. R. M. B. M. Farsi, "An Overview of Controller Area Network," IET Digital Library.

[8] Spark Fun, "Serial Communcation," 2015. [Online]. Available: https://learn.sparkfun.com/tutorials/serial-communication/all.

[9] "Serial Communication," [Online]. Available: https://en.wikipedia.org/wiki/File:Parallel_and_Serial_Transmission.gif.

[10 F. Malpartida, "LCD Display," [Online]. Available: https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/Home.

[11 Omerk, "LCD Chargen," [Online]. Available: https://omerk.github.io/lcdchargen/.

[12 M. Balboa, "RFID," [Online]. Available: https://www.github.com/miguelbalboa/rfid.

[13 Arduino LLC, "Arduino Uno," [Online]. Available: https://www.arduino.cc/en/Reference/Board.

Report Source Analysis

Source	Credibility		Information	
	Reputation and Expertise	Reliability (out of 10)	Further Research / Verification	Key Findings
1] W. D. D.Hahnel, "Mapping and localization with RFID echnology," Freiburg University.	Dirk Haehnel is a google developer who has been publishing papers since 1999 and is cited up to 1000 times a year according to the researchers Google Scholar profile.	9	Further research from the university of west Virginia (http://rfid.cs.virginia.edu/work.pdf) summarises the physics of this technology and further expands on its application.	The paper outlined the practical applications o RFID technology in the localisation of objects.
2] Arduino LLC, "What is arduino?," 2015. [Online]. Available: ıttps://goo.gl/Sso1gH	Arduino LLC specialises in building micro-controllers for prototyping and consumer based DIY projects.	10	An article from "makeuseof.com" (www.makeuseof.com/tag/getting-started-with-arduino-a-beginners-guide/) provided no contradictory evidence on the matter, however it did expand upon the specification of the boards and which variants were compatible	The paper outlines the key tools and knowledge needed to program and flash thei microcontrollers.
3] Arduino LLC, "SD Library Documentation," [Online]. Available: ıttps://www.arduino.cc/en/Reference/SDbegin.	[Source 2]	10	I further delved into the practical application of the module and found an interesting article of live data logging (https://howtomechatronics.com/tutorials/arduino/arduino-sd-card-data-logging-excel-tutorial/). It also outlines the key commands for the module and the library required.	I found the wiring of the module, its power requirements and the key commands needed
4] Arduino LLC, "LCD Crystal library," [Online]. Available: ıttps://www.arduino.cc/en/Reference/LiquidCrystal.	[Source 2]	6	I have rated the following source with a low 6 due to problems I encountered with the print function of that particular library, I found another library that seemed to be more reliable https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/Home).	[Source 3]
5] Arduino LLC, "MFRC522," 2013. [Online]. Available: ıttps://playground.arduino.cc/Learning/MFRC522.	[Source 2]	10	Source 5 mentioned the specific protocol used to transfer data from the module to the main controller and further research into this proved quite beneficial (refer to source 9). Further research also brought up a useful library (see source 12).	[Source 3] & I was also informed on the compatibility of various Mifare cards.

Report Source Analysis

6] Arduino LLC, "Arduino Mega 2560," Arduino, [Online]. Available: https://www.arduino.cc/en/Main/arduinoBoardMega2560/.	[Source 2]	10	The documentation supplied from this source was exemplary and I found no contradictions.	Supplied evidence of voltage requirements for the module, current limitations and over all specifications.
7] K. R. M. B. M. Farsi, "An Overview of Controller Area Network," IET Digital Library.	The paper was a collaborative effort from three authors from Newcastle upon Tyne. Dr M. Farsi the main author of the paper has published quite a few papers around the subject and seems to be very knowledgeable on the subject.	9	I also found another paper written by Jose Rufino confirming the findings of this paper (http://dario.di.fc.ul.pt/downloads/ciaBrg.pdf).	Although the term Can bus is usually used in the car industry, the same techniques are also applicable to closed systems such as RFID door lock controls The paper also provided and insight into the Synchronous transmission of packets and collision detection essential for understanding the SPI interface
8] Spark Fun, "Serial Communcation," 2015. [Online]. Available: https://learn.sparkfun.com/tutorials/serial-communication/all.	Similar to source 2	7	I furthered my research into logic levels which is a fundamental component of this protocol (https://learn.sparkfun.com/tutorials/logic-levels).	Provided an insight into where the protocol in the context of its application in Arduino based projects such as mine.
9] "Serial Communication," [Online]. Available: https://en.wikipedia.org/wiki/File:Parallel_and_Serial_Transmission.gif.	There is no confirmative evidence to either verify or not verify the credentials of the source which is why I have confirmed it with a wide range of others.	6	Supporting Sources: http://slideplayer.com/slide/6652294/ http://www.newhavendisplay.com/serialvsparallel.html https://epqbvwilson.wordpress.com/	The image provided a visual differentiation between Serial and Parallel lines.
10] F. Malpartida, "LCD Display," [Online].	F. Malpartida's efforts were published on a	8	This is an extension from my research into source 3 which revealed a fault in the Arduino library for the LCD. Further	This library provided a functioning print

Report Source Analysis

Available: https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/Home.	collaborative site where his work is constantly verified by a community of users.		research showed that it was possible to create custom characters and I found this useful tool to do this which output the raw code (Source 10)	function and documentation on creating custom characters for the project.
11] Omerk, "LCD Chargen," [Online]. Available: https://omerk.github.io/lcdchargen/.	No Verification was needed as his Tool worked perfectly.	10	No further research was required.	Provided the exact format for creating a bit array for coding custom characters.
12] M. Balboa, "RFID," [Online]. Available: https://www.github.com/miguelbalboa/rfid.	Similar to source 10	8	Further research revealed the protocol that the library used to communicate with the board (SPI). For more see the US patent for this technology: (https://patents.google.com/patent/US4742349A/en).	Provided key commands for extracting, verifying and also writing UIDS from the RFID chips.
13] Arduino LLC, "Arduino Uno," [Online]. Available: https://www.arduino.cc/en/Reference/Board.	[Source 2]	10	This was an adequate source of information.	Provided specs on the different Arduino boards.

Artefact and some additional sources:

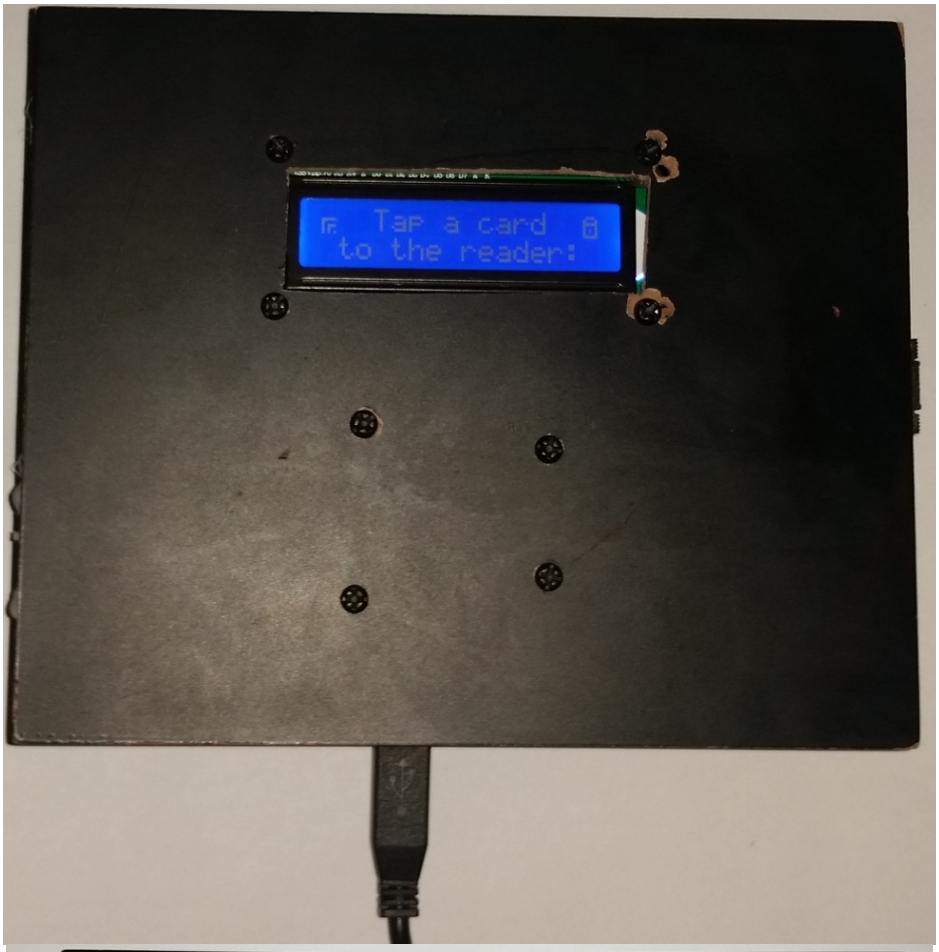
1. "diyelectricskateboard" [online] - http://www.diyelectricskateboard.com/diy-electric-skateboard-kits-parts/torqueboards-dual-motor-mechanical-kit/
2. "runplayback" [online] -http://www.runplayback.com/how-i-built-a-diy-eboard
3. "wikipedia" [online] - https://en.wikipedia.org/wiki/Electronic_speed_control
4. "rancidbacon" [online]- http://www.rancidbacon.com/files/kiwicon8/ESP8266_WiFi_Module_Quick_Start_Guide_v_1.0.4.pdf
5. "arduino" [online] - https://www.arduino.cc/en/reference/serial
6. "arduino" [online] - http://playground.arduino.cc/Interfacing/Java
7. "ledsupply" [online] - http://www.ledsupply.com/blog/why-you-need-an-led-heat-sink/
8. "amazon" [online] - https://www.amazon.co.uk/UEB-MFRC-522-Radiofrequency-Inducing-Arduino/dp/B01D168X9Y/ref=sr_1_1?ie=UTF8&qid=1490108102&sr=8-1&keywords=rc522
9. "github" [online] - https://www.github.com/miguelbalboa/rfid
10. "oscarliang" [online] - https://www.oscarliang.com/sd-card-arduino/
11. "github" [online] - https://www.github.com/miguelbalboa/rfid
12. "wikipedia" [online] -- https://en.wikipedia.org/wiki/Serial_communication

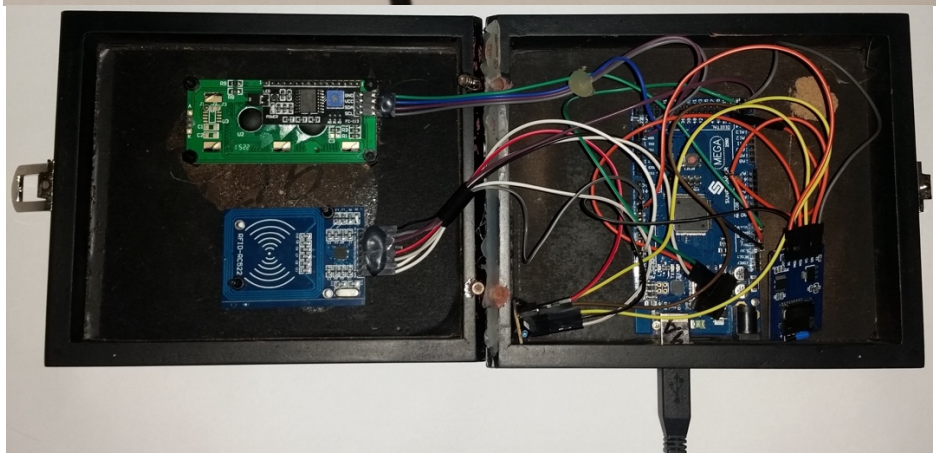
Report Source Analysis

13. "wikipedia" [online] - https://en.wikipedia.org/wiki/File:Parallel_and_Serial_Transmission.gif
14. "wikipedia" [online] - https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus
15. "arduino" [online] - https://www.arduino.cc/en/Reference/SDbegin
16. "arduino" [online] - https://www.arduino.cc/en/reference/SD
17. "adafruit" [online] - https://learn.adafruit.com/i2c-spi-lcd-backpack/overview
18. "wikipedia" [online] - https://en.wikipedia.org/wiki/Microcontroller
19. "arduino" [online] - https://www.arduino.cc/en/reference/wire
20. "arduino" [online] -- http://playground.arduino.cc/Main/InterfacingWithHardware
21. "electroschematics" [online] - http://www.electroschematics.com/12459/arduino-i2c-lcd-backpack-introductory-tutorial/
22. "github" [online] - https://omerk.github.io/lcdchargen/
23. "impinj" [online] - http://www.impinj.com/resources/about-rfid/the-different-types-of-rfid-systems/
24. "gadgetronicx" [online] - http://www.gadgetronicx.com/interface-rfid-with-avr-microcontroller/
25. "sogeti" [online] - http://esec-pentest.sogeti.com/posts/2011/11/28/playing-with-nfc-for-fun-and-coffee.html
26. "arduino" [online] - https://forum.arduino.cc/index.php?topic=79035.0
27. "sunfounder" [online] - http://www.sunfounder.com/wiki/index.php?title=I2C_LCD1602
28. "ebay" [online] - http://www.ebay.co.uk/itm/UID-card-Changeable-Writable-Rewritable-MF1-IC-13-56Mhz-thin-card-For-test-5pcs-/181968592146?hash=item2a5e2c5d12:g:e48AAOSwk1JWd6HA
29. "eng" [online] - http://www.eng.utah.edu/~cs3992/archive/2008/RFID-final-prop.pdf

Final Artefact

Photos of the Artefact






Allington Tech


Admin Mode
Add / Remove
Students' union


05 Added Card
2ed750cd
ST BRENDAN'S
SIXTH FORM COLLEGE
Mathew Allington
Student ID:160471
Email:160471@student.stbrn.ac.uk
EXPIRY DATE:20/06/2018


04Removed Card
2ed750cd
ST BRENDAN'S
SIXTH FORM COLLEGE
EXPIRY DATE:20/06/2018

Video of the Artefact

EPQ RFID artefact demonstration (RFID door lock)

OR

www.youtube.com/watch?v=MTY0NvSVSqE

Artefact Code for Arduino Mega 2560

```
/*
* Author : Mathew Allington
*/
//LCD Crystal Display Variables & Imports
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2); // Sets
the dimensions of the display
byte timerChar[8] = {
0b00000,
0b01110,
0b10011,
0b11101,
0b10001,
0b01110,
0b00000,
0b00000
};
byte signalChar[8] = { //Bitmap created by
generator :
https://omerk.github.io/lcdchargen/
0b00000,
0b00000,
0b00000,
0b11111,
0b10000,
0b10111,
0b10100,
0b10101
};
byte openChar[8] = {
0b11111,
0b11001,
0b11101,
0b11101,
0b11101,
0b11111,
0b01100,
0b00100
};
byte lockedChar[8] = {
0b00000,
0b01110,
0b10001,
0b11111,
0b10001,
0b10101,
0b10001,
0b11111
};
byte deniedChar[8] = {
0b10001,
0b01010,
0b01010,
0b00100,
0b00100,
0b01010,
0b01010,
0b10001
};
byte adminChar[8] = {
0b00000,
0b11111,
0b10001,
0b10001,
0b11111,
0b01110,
0b11111,
0b00000
};
byte removedChar[8] = {
0b00100,
0b11111,
0b00000,
0b11111,
0b10101,
0b10101,
0b10101,
0b01110
};
byte addedChar[8] = {
0b11111,
0b10001,
0b10101,
0b11111,
0b10101,
0b10001,
0b11111,
0b00000
};
byte splashChar[8] = {
0b10001,
0b01010,
0b10101,
0b01010,
0b10101,
0b01010,
0b10101,
0b01010
};
byte errorChar[8] = {
0b00100,
0b01110,
0b01010,
0b11011,
0b11111,
0b11011,
0b11111,
0b00000
};
#if defined(ARDUINO) && ARDUINO >= 100
#define printByte(args)  write(args);
#else
#define printByte(args)  print(args,BYTE);
#endif
int currentIcon = 1;
//------------------------------
//RFID DoorLock Variables & Imports
int doorPin = 7;
//RFID Reader Variables & Imports
String masterKey= String("de9d54cd");
#include <SPI.h>
#include <MFRC522.h>
//#define SS_PIN 10
#define RST_PIN 3 // Changed to 3 because of
conflict with SD module
MFRC522 rfid(10, RST_PIN);
MFRC522::MIFARE_Key key;
byte currentCard[4];
//------------------------------
// SD Card Reader Variables & Imports
#include <SD.h>
```

```
String* storedUIDs;//[2]=
{"2ed750cd","fee04364"};
File myFile;
int UIDsAmount =0;
String UIDLocation = "UIDs.txt";
int SDPin = 4; // this is the pin the SD
starts up on
//------------------------------
//Main program variables
int MODE = 0;
int NORMAL = 0;
int ADMIN_SNIFFER = 1;
//--------------------
void setup() {
setUpLock();
setLockOn(true);
Serial.begin(9600);
Serial.println("Lock on");
setUpLCD("","");
Serial.println("Lcd on");
splashScreen(); // Boot sequence
Serial.println("Splash Screen up");
setUpReader();
initSD();
storedUIDs = loadUIDs();
printArray(storedUIDs, UIDsAmount);
Serial.println(String("Gathered")+=String(UI
DsAmount)+=String(" UIDs"));
//if(!(setUpSD())){
// setUpSD();
//while(true);
//}
homeScreen();
}
void loop() {
switch(MODE){
case 0:
if(cardPresent()){
Serial.println("Card Detected");
Serial.println("New Card!");
readCard();
String cardContents =
byteToHex(currentCard,4);// need to convert
to hexadecimal string!!!
Serial.println(cardContents);
if(idEquals(cardContents, masterKey)) {
adminScreen();
Serial.println("Waiting for new card");
delay(2000);
MODE = 1;
}
else if(authID(cardContents)){
setLockOn(false);
// errorScreen(cardContents);
unlockedScreen();
timer(5);
setLockOn(true);
homeScreen();
}
else{
deniedScreen();
Serial.println(cardContents);
timer(5);
homeScreen();
}
}
break;
case 1:
Serial.println("Looking for new card");
adminSniffer();
break;
}
}
// Main program methods
void adminSniffer(){
if(cardPresent()){
delay(1000);
readCard();}
String toAdd = byteToHex(currentCard,4);
if(toAdd.equals(masterKey)) return;
if(authID(toAdd)){
Serial.println(String("Attempting to remove
card:")+=toAdd);
removeAll(toAdd);
removedScreen(toAdd);
writeUIDs(storedUIDs, UIDsAmount);
Serial.println("Reloading");
storedUIDs = loadUIDs();
timer(5);
homeScreen();
MODE =0;
}
else{
Serial.println(String ("Adding new UID:
")+toAdd);
addedScreen(toAdd);
addUID(toAdd);
Serial.println("Reloading");
storedUIDs = loadUIDs();
timer(5);
homeScreen();
MODE =0;
}
}
//------------------------
void removeAll(String syntax){
for(int i = 0; i< UIDsAmount; i++){
if(idEquals( storedUIDs[i], syntax)){
storedUIDs[i] = "";
}
}
}
//LCD Crystal Display Methods
void transition(int timePer){
for(int i = 0; i<=16;i++){
delay(timePer);
lcd.scrollDisplayLeft();
}
}
void testScreen(int wait){
addedScreen("TEST");
delay(wait);
removedScreen("TEST");
delay(wait);
adminScreen();
delay(wait);
unlockedScreen();
delay(wait);
deniedScreen();
delay(wait);
homeScreen();
delay(wait);
}
void errorScreen(String reason){
cleanPrint("Error",reason, true);
printIcon(15,0,errorChar);
printIcon(0,0,errorChar);
}
void addedScreen(String UID){
cleanPrint("Added Card",UID, true);
```

```
printIcon(15,0,addedChar);
}
void removedScreen(String UID){
cleanPrint("Removed Card",UID, true);
printIcon(15,0,removedChar);
}
void adminScreen(){
cleanPrint("Admin Mode","Add /
Remove",true);
printIcon(15,0,adminChar);
}
void unlockedScreen(){
cleanPrint("Accepted,","now enter!",true);
printIcon(15,0,openChar);
}
void deniedScreen(){
cleanPrint("DENIED,","invalid key.",true);
printIcon(15,0,deniedChar);
}
void homeScreen(){
cleanPrint("Tap a card","to the
reader:",true);
printIcon(0,0,signalChar);
printIcon(15,0,lockedChar);
}
void splashScreen(){
printIcon(0,0,splashChar);
printIcon(0,1,splashChar);
printIcon(15,0,splashChar);
printIcon(15,1,splashChar);
for(int i=  0; i<=16; i++){
String toPrint = String("");
for(int j =0; j<i; j++){
toPrint += "_";
}
cleanPrint("Allington Tech",toPrint,false);
delay(50);
}
delay(2000);
}
void timer(int period){
printIcon(0,0,timerChar);
lcd.setCursor(1,0);
for(int i=period; 0<i;i--){
lcd.setCursor(1,0);
lcd.print(String(" "));
lcd.setCursor(1,0);
lcd.print(String("")+=i);
delay(1000);
}
}
void printIcon(int collum, int line, byte
icon[]){
lcd.createChar(currentIcon, icon);
lcd.home();
lcd.setCursor(collum, line);
lcd.printByte(currentIcon);
currentIcon++;
if(currentIcon>12) currentIcon = 1; //
because of char limit it resets the number
}
void setUpLCD(String header, String
message){
lcd.init();
//lcd.begin(16,2);
lcd.backlight();
cleanPrint(header, message,true);
}
void cleanPrint(String header, String
message,bool clearLCD){
if(clearLCD){
lcd.clear();}
lcd.setCursor((16-header.length())/2, 0);
lcd.print(header);
lcd.setCursor((16-message.length())/2, 1);
lcd.print(message);
}
//------------------------------
//Door Lock Methods
void setUpLock(){
pinMode(doorPin, OUTPUT);
}
void setLockOn(boolean on){
if(on){
digitalWrite(doorPin, HIGH);
}
else{
digitalWrite(doorPin, LOW);
}
}
//-------------------------
//RFID Reader Methods
void setUpReader(){
SPI.begin(); // Init SPI bus
rfid.PCD_Init(); // Init MFRC522
for (byte i = 0; i < 6; i++) { //
Initializes the byte array for the key
variable
key.keyByte[i] = 0xFF;
}
}
boolean cardPresent(){
if ( ! rfid.PICC_IsNewCardPresent())
return false;
// Verify if the NUID has been readed
if ( ! rfid.PICC_ReadCardSerial())
return false;
}
boolean cardTypeValid(){
MFRC522::PICC_Type piccType =
rfid.PICC_GetType(rfid.uid.sak);
// Check is the PICC of Classic MIFARE type
if (piccType !=
MFRC522::PICC_TYPE_MIFARE_MINI &&
piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
return false;
}
return true;
}
boolean newCard(){
return (rfid.uid.uidByte[0] !=
currentCard[0] ||
rfid.uid.uidByte[1] != currentCard[1] ||
rfid.uid.uidByte[2] != currentCard[2] ||
rfid.uid.uidByte[3] != currentCard[3] );
}
void readCard(){
for (byte i = 0; i < 4; i++) {
currentCard[i] = rfid.uid.uidByte[i];
}
}
//------------------------------
// SD Card Reader Methods
void initSD(){
if (!SD.begin(4)) {
Serial.println("initialization failed!");
return;
}
Serial.println("initialization done.");
```

Implementation Diary / Log

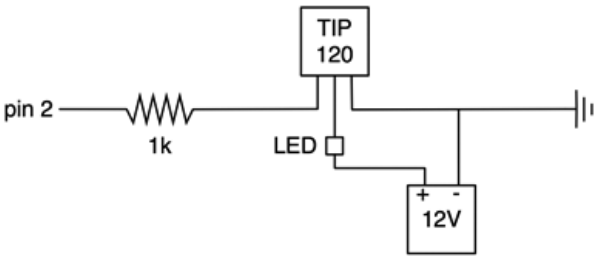
Date	Action	References
22/7/17	I searched online for different Arduino projects and came up with these ideas. • Home Automation: This entails the use of relays, door locks, RFID readers and LCD crystal displays. • RGB Flood light: makes use of a 100w RGB SMD which would be controlled through PWM to get different colour outputs. • Electric skateboard- Using the microcontroller to generate a pulse that would be controlled by a Bluetooth module, the pulse would hen control the speed of the motor.	Home Automation: https://youtu.be/MnaryeiBStM RGB Flood Light: https://youtu.be/c--5c3Egv4E https://youtu.be/BCQq7xRX-L8 Electric Skateboard: https://youtu.be/7b2o2GhqsoQ
05/8/17	I researched my three project ideas in more depth so that I could refine them to a final idea. For this I spoke to an electrical engineer idea (Mike Allington) who consulted me on my RGB flood light (idea 3). I chose idea 1 (NFC Door lock) and evaluated my choice in the project proposal where I did research on the more particular components of the design.	http://playground.arduino.cc/Learning/PRFID https://www.amazon.co.uk/UEB-MFRC-522-Radiofrequency-Inducing-Arduino/dp/B01D168X9Y/ref=sr_1_3?ie=UTF8&qid=1481018239&sr=8-3&keywords=rfid+arduino
6/1/18	After my modules arrived in the post, I started researching how the NFC, SD and LCD crystal display modules operated. I then started to implement them: **SD Module –** The SD pin setup was pretty straight forward, because I used a diagram that I found online (link 1). I then used an SD library (included with IDE) and one of the example sketches to test that the module worked. I then wrote to a text file with the syntax "myFile.write()" and then read the file as well . **NFC Module –** I wired the module up using the suggested pins and used the recommended "Miguelbalboa" library (link 2) to read a card. After a bit of research (link 3), I learned that NFC and RFID cards are broken up into sectors. Sector 0 stores what is called a Unique Identification (UID). This UID can be read and then used to validate the card. This means I now have a way to differentiate between door access cards. **LCD Crystal Display –** The display I used has 16*2 characters. Pin setup only involved 4 pins. **GND,VCC** – for 5V power, **SDA –** goes to A4 on the Arduino, **SCL** – goes to A5 on the Arduino. I then setup up the LCD with the syntax "lcd.init()" and "lcd.backlight()" to turn the back light on. I then used the "lcd.println()" command to print "Hello, World" on the screen.	**Link 1 -** http://www.14core.com/wp-content/uploads/2015/10/Wiring-SD-Card-Module-Pinout-Schematics-diagram-Arduino-UNO.jpg **Link 2-** https://github.com/miguelbalboa/rfid **Link 3 -** https://firefart.at/post/how-to-crack-mifare-classic-cards/

	I then paired this with the NFC module in a new sketch to test that the modules could work simultaneously.	
25/1/18 Objectives –Getting the modules to work simultaneously (✓) -Creating the display methods and the custom characters (✓s).	**11:00 -** I started this preliminary session by adding the connecting all of the modules together in order to ensure that they could work simultaneously. However, I came across a **problem number 1**, which meant the SD would not initialise properly. I found out that this was due to the "RST" pin on both the SD and RFID module. The SD module would not initialise because they were using the same pin (pin 4). So, I **found a fix** using one of the example sketches on the "Miguelbalboa" library, I found out that you can change this pin using the line of code "#define RST_PIN 3 ". I found out that this redefines the global variable in the library. **13:00 - Problem number 2** – when testing the LCD display, I found that when printing a line, it would always be followed by an un- intentional character. My suspicion was that this was due to a bug in the library I was using. **To fix this** I found the manual (Link 1) for a similar module and used a different library. **14: 00 -** After overcoming these problems, created the methods "cleanPrint" and "setupLcd." The clean print method allows me to print multiple lines with ease and in a clean format. **15: 00 -** I then started to create custom chars that I could output to the screen. I did this by creating a binary array, which would define each individual pixel in each character set. I found a great website that I could create custom chars and then output its binary array form (Link 2). Using this website I created a denied, open, admin, signal and locked icon. I then used these with different screens that I bunched up into methods that I could call for different stages of the flow chart that I made on my project proposal.	**Link 1 (Manual LCD) -** http://www.sunfounder.com/wiki/index.php?title=I²C_LCD1602 **Link 2 (custom chars) –** https://omerk.github.io/lcdchargen/
28/1/18 Objectives -Getting the RFID module to read cards (✓). - Saving and storing the cards on an SD module (✘).	After creating all of the graphics screens for the display, I then created the methods for the MFRC522 RFID reader module. This included initialising the module and methods that checked if there was a card near by, the validity of the card etc. I did this by analysing the example sketches that came with the module's libraries: 8:00 – From preliminary testing on the 6th of January, I	

	found that there are two pins that can be changed so I defined these in a global scope so that the MFRC522 class could access it. I did this using the “#define” syntax. 9:00 – After further debugging, I found that to initialise the module you have to make a new instance of the class then you call the non- static method “PCD_Init()”. I put this in a simple method (“setUpReader()”) along with a for loop that initialises a byte array that I will later use to store the card’s UID in. 9:30 – I needed a way of gaging whether there is a card to read, so I found a statement that returns a Boolean, it would return “true” for a card present and false for no card at all. I then put this into a method that also checked if the data line that the module communicated on was available for use. I then returned a Boolean from this method then used it as a condition in an if statement. This was implemented in the loop section so that it would constantly iterate round and then check to see if there was a card. 10:30 – I then attempted to implement the SD library into my project using the “#include” syntax. Also initially the compilation was okay, when the Arduino ran it stopped serial response all together. To try and find the source of the problem I used “Serial.println()” with the appropriate string as a parameter. I used in the setup() method to see how far the Arduino got before crashing completely. I found that this was while the SD card module was being initialized. I found that this was due to the dynamic memory being filled up. This was because the module only supported up to 2kb. So naturally, I upgraded the module to an Arduino Mega that has 8kb of dynamic memory. 19:00 – After the module arrived I did some quick testing using the LCD Crystal display. I found that the pins (20,21) that the display communicated on had a hairline fracture on the board leading to them. So, as a result the module did not work. So this mean that the objective of saving the UIDs to an SD was not met for the day. So I ordered another board for the next day.	

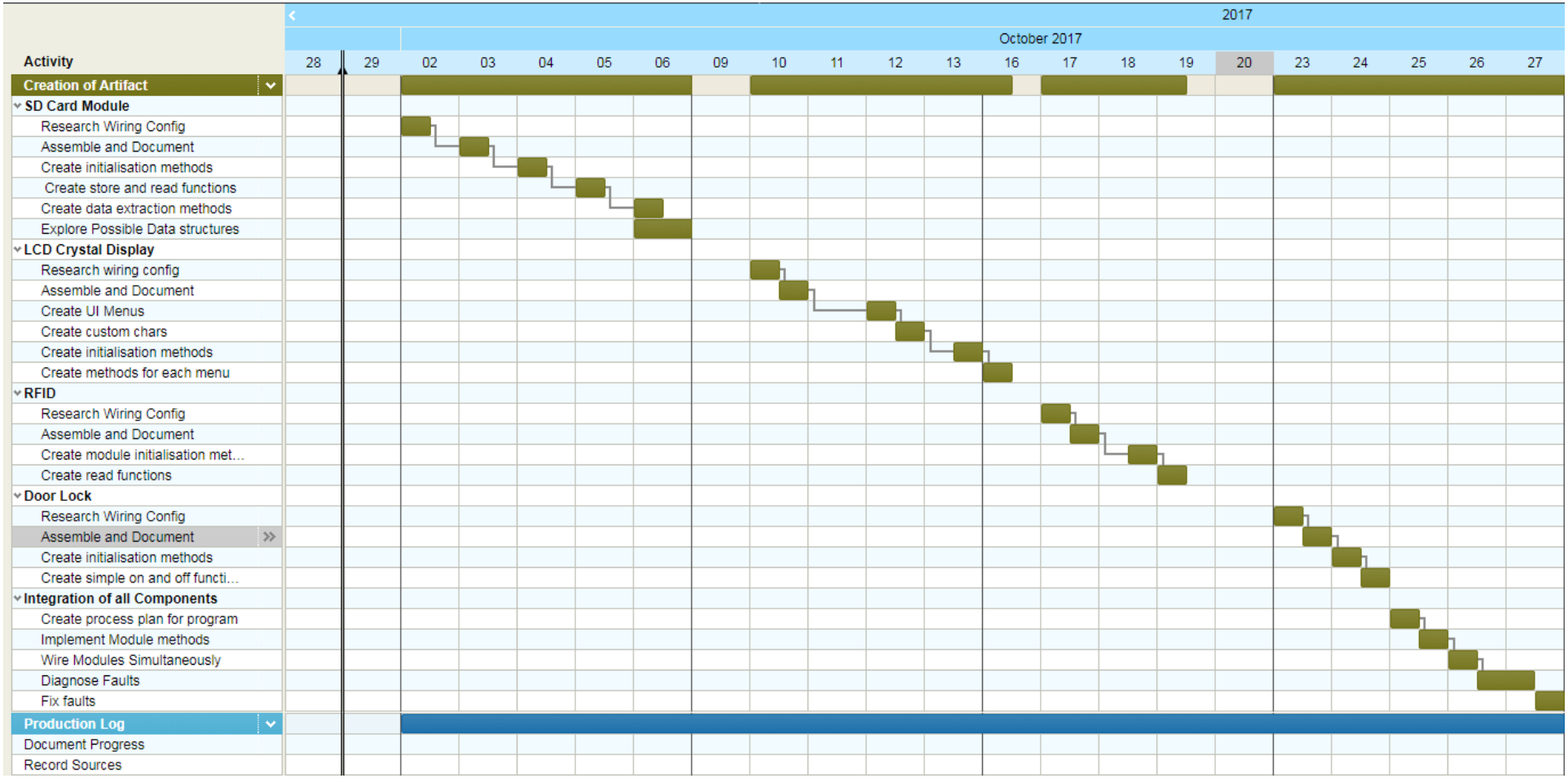
29/1/18 Objectives - (**continued on**) Saving and storing the cards on an SD module (✓).	18:00 – The new second Arduino mega arrive I quickly tested it with the LCD Screen and found that it in fact worked. 18:30 – I then attempted to implement the SD library and confirmed that it wasn't occupying any dynamic space. 19:00 - I then started analysing example sketches that came with the SD module. I found to use the SD module you have to create a new instance of the "File" class. This allowed me to then initialise the SD card and then set the directory that I would be working in. 20:00 – After getting to know the basics I started testing in a blank sketch while putting them into methods that I could then use in the real thing. 20:20 – First I started implementing the initialise function, I returned a Boolean from this function so that I could verify that the module was working at start up. Otherwise it would display an error screen which would prompt the user to check the SD connections. 20:40 – After I was satisfied with that, I started writing the read and write functions for my project. I started by creating some global variables that I could call directly from the sketch itself. This made it easier than inputting the parameters into the methods directly as the code looks a lot cleaner. These variables included a string which stored which directory the SD module is reading and writing from. Also a Boolean that states whether the module is initialised. In addition I included a String array that I would later use to store the UIDs and an Integer that stated exactly how many UIDs were in the array at any present time. 21:00 – After I fully implemented the methods I started writing methods that could extract the data into a clean String array. That I could use in the next implementation day. Also, other methods that can rewrite text data into a CSV format so that it can be successfully re-imported from the file.	
1/2/18 Objectives - To successfully extract UIDs from a CSV file (✓). - Use this data to verify a correct UID at the door (✓). - To be able to add	10:00 – I started off by continuing the process of writing a system that can extract and store the UIDS. I devised a system where the "String* loadUIDs()" method iterated through the raw data from the read methods. When it came across a ', ' character it would ignore it and iterate through the array. This process would repeat until all of the characters are complete. 12:00 – After I'd done that I created a method that	

and remove UIDs (✓).	rewrites the text file when the contents of the UID[] array had changed. However, I came across a problem, when the text file was rewritten it would only rewrite to the end of the file and therefore it would just append to it. I looked through the documentation for the library and I didn't really find anything, however I could dispose of the file. So I used the "SD.remove(String Directory)" from the library so that it would be starting a new text file when it re-wrote the UIDs. 14:00 – After that I started writing the methods that would verify and check the UIDs against the array. I did this by first taking in the UID if it was available. I did this by using the RFID methods that I had created earlier on. Once the program had obtained the byte array it would then check against the array. However, because the inputted data from the card was a byte, it was slightly difficult. So in order to compare the values as one data type I had to convert it to a hexadecimal string (eg. "E4H6C6"), I did this by initialising a new instance of a string object via the method "string ident = String(byteArray, "HEX");". The "HEX" parameter states the format that the string would be stored in.	
2/2/18 Objectives - To successfully be able to authenticate a key card against the CSV file (✓).	10:00 – In order to authenticate a card when presented, I created a method that iterated through the array and checked if any of the strings equalled to the hexadecimal equivalent of the cards byte array. However, I came across a problem, when I entered a valid card into the array, it didn't seem to be validated when a card was presented. After some debugging, I found that when the programming was writing a UID into the array, it would also enter escape characters and other different string carriage return types. In order to fix this, when I was reading from the array, I checked to see if the value of the char was within a certain range. This would be the range in which the normal hexadecimal string characters would lye. So this means it would ignore all of the excess characters. 14:00 – My next task to complete was the operation of the door lock. The next problem I faced was the voltage requirements of the lock. This is because the lock required 12v and 1A of current. This meant that I could not use the standard pins from the Arduino, because this would only produce 3.3v and 10mA of current. In order to be able to operate the lock via these pins I used a TIP120 transistor to turn a 12v circuit on and off when the pin was high and low.	**Diagram:** http://www.mit.edu/~kimo/blog/leds.html

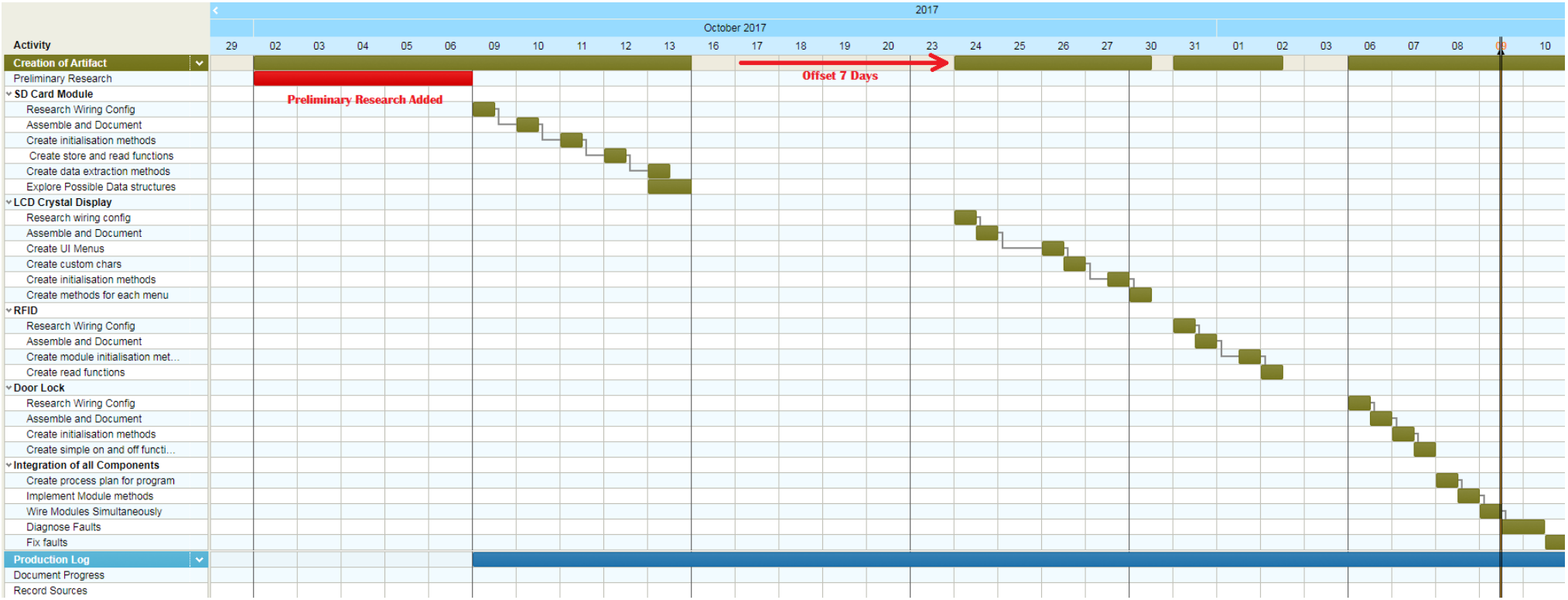
	 Using this diagram, I simply just replaced the led module for the electromagnetic door lock.	
3/2/18 <u>Objectives</u> - To operate the on/off function on the door lock (✓).	10:00 – After I had soldered the power board for the lock, I created a simple "lock();" method that set the pin state to high when the no card was presented and then "unlock();" when a valid card was is proximity 1:00 – I then made some improvements, including the pin state of the door lock when rebooting the module. The problem occurred when I reset the sketch with the reset button. The door lock would be unlocked for significant amount of time while the setup was reading and parsing the UIDs from the SD. To solve this problem, I just made the lock instantly initiate as soon as the controller was up.	

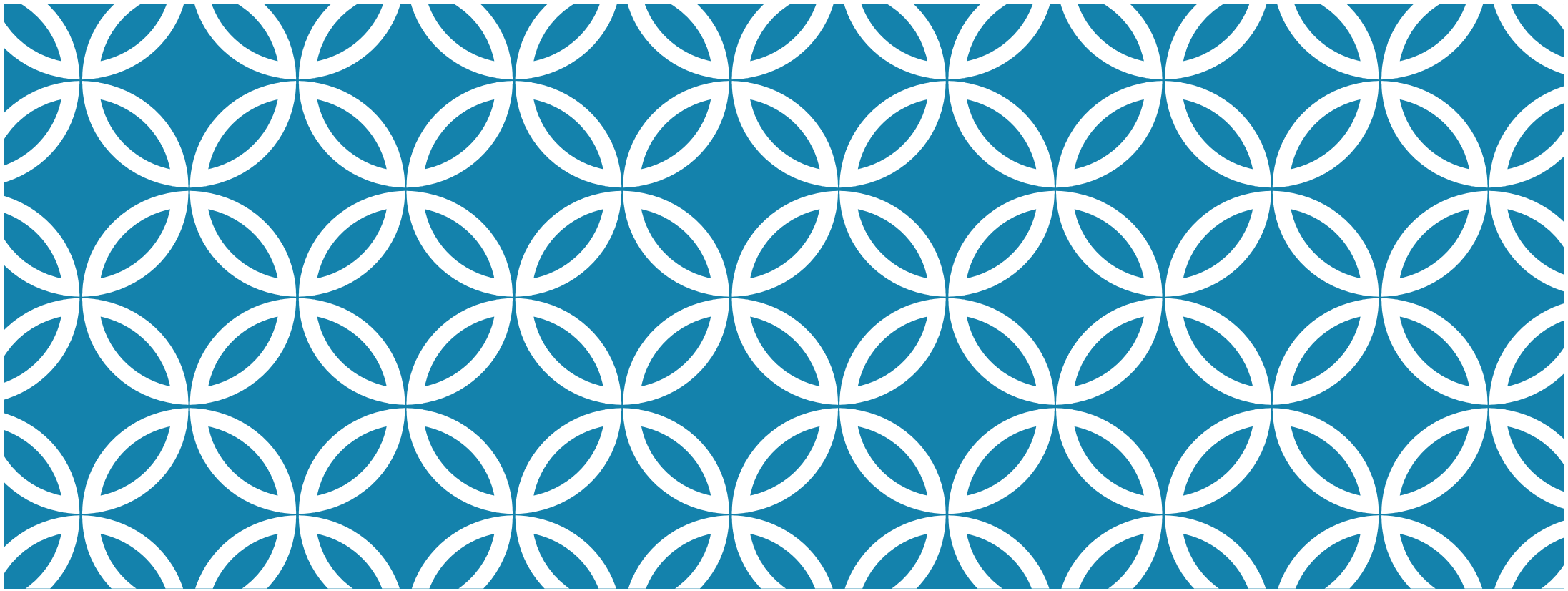
Gantt Chart

Version 1



VERSION 2


2017
October 2017
Activity
29 02 03 04 05 06 09 10 11 12 13 16 17 18 19 20 23 24 25 26 27 30 31 01 02 03 06 07 08 09 10
Creation of Artifact
Preliminary Research
SD Card Module
Research Wiring Config
Assemble and Document
Create initialisation methods
Create store and read functions
Create data extraction methods
Explore Possible Data structures
LCD Crystal Display
Research wiring config
Assemble and Document
Create UI Menus
Create custom chars
Create initialisation methods
Create methods for each menu
RFID
Research Wiring Config
Assemble and Document
Create module initialisation met...
Create read functions
Door Lock
Research Wiring Config
Assemble and Document
Create initialisation methods
Create simple on and off functi...
Integration of all Components
Create process plan for program
Implement Module methods
Wire Modules Simultaneously
Diagnose Faults
Fix faults
Production Log
Document Progress
Record Sources
Offset 7 Days
Preliminary Research Added

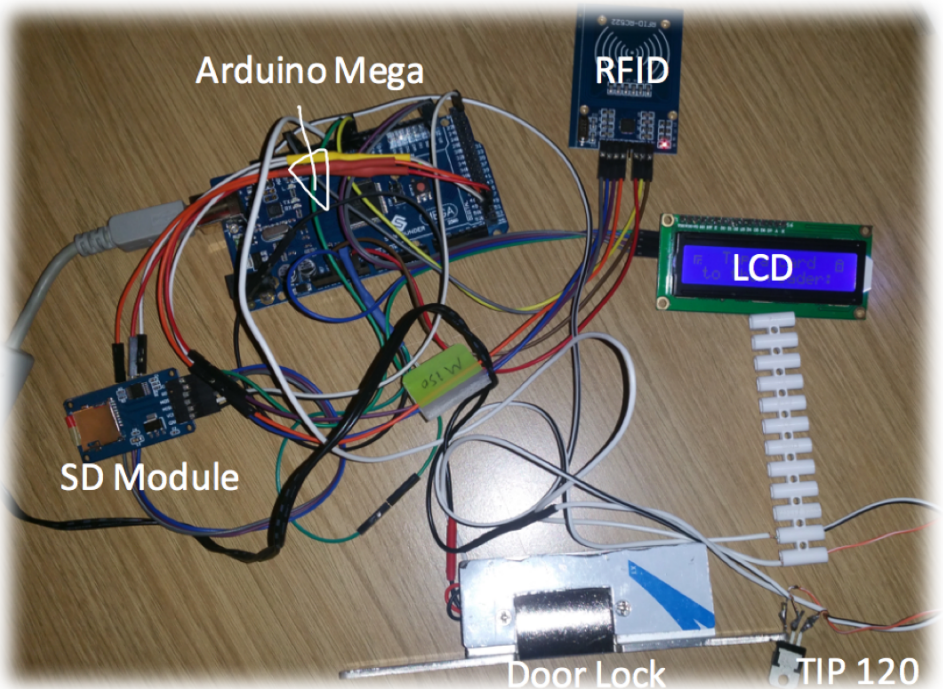


BUILDING AN RFID DOOR LOCK

Author – Mathew Allington

MY PROJECT


Tap a card
to the reader:


Arduino Mega
RFID
LCD
SD Module
Door Lock
TIP 120

MY INTERESTS

```
+Sector: 0
FEE0436439880400C834002000000015
00000000000000000000000000000000
00000000000000000000000000000000
FFFFFFFFFFFFFF078069FFFFFFFFFFFF
+Sector: 1
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
FFFFFFFFFFFFFF078069FFFFFFFFFFFF
+Sector: 2
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
FFFFFFFFFFFFFF078069FFFFFFFFFFFF
```

Cloning Cards

Home Automation

Security Exploits

Knowledge Expansion

MY AIMS AND OBJECTIVES

1. To successfully create a system in which I can add remove users
2. Store UID numbers on an SD module or internal Arduino
3. Get the RFID module to read UIDs
4. Get the magnetic door lock to open and close

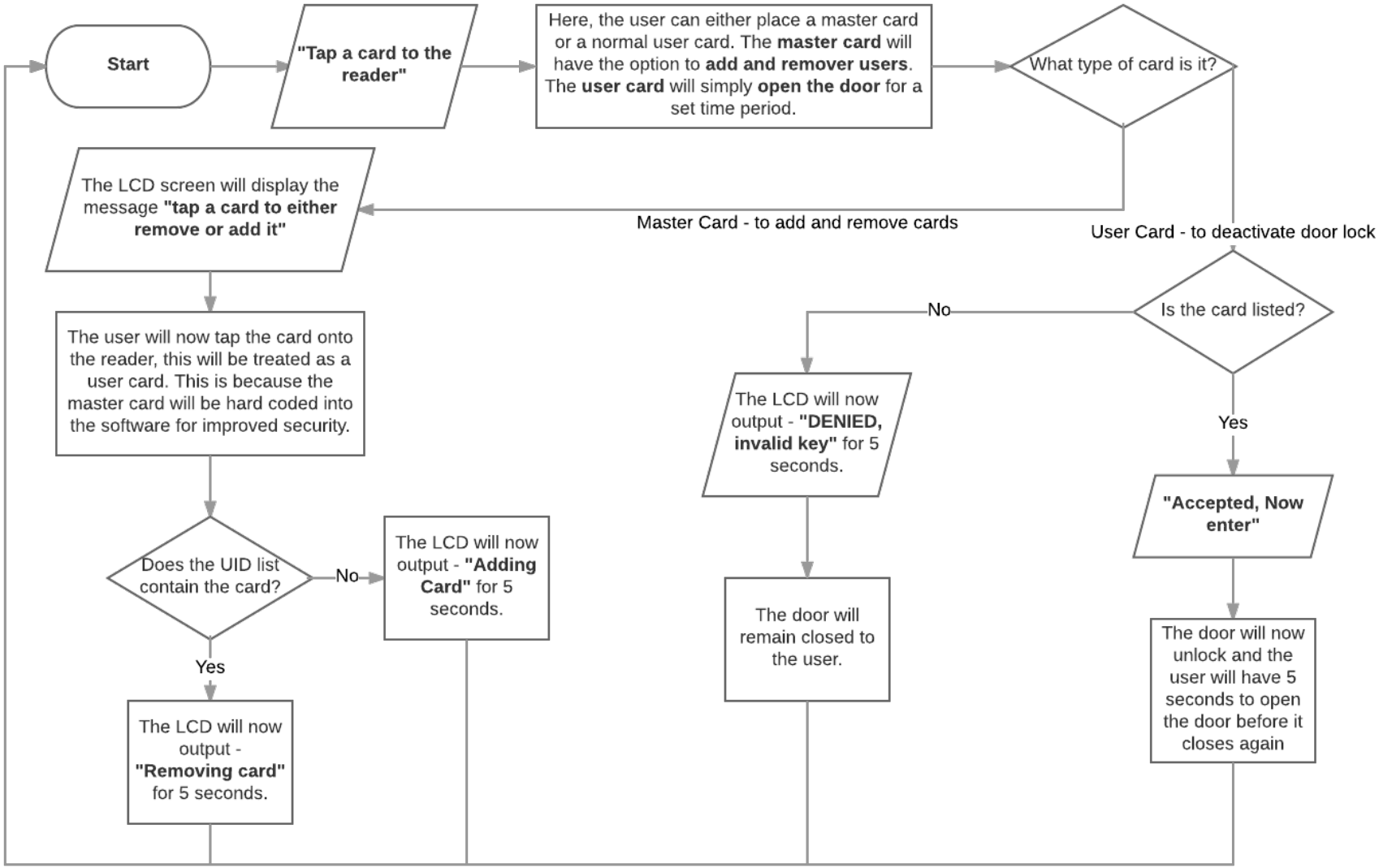
Card recognition

User Interface

Memory Storage

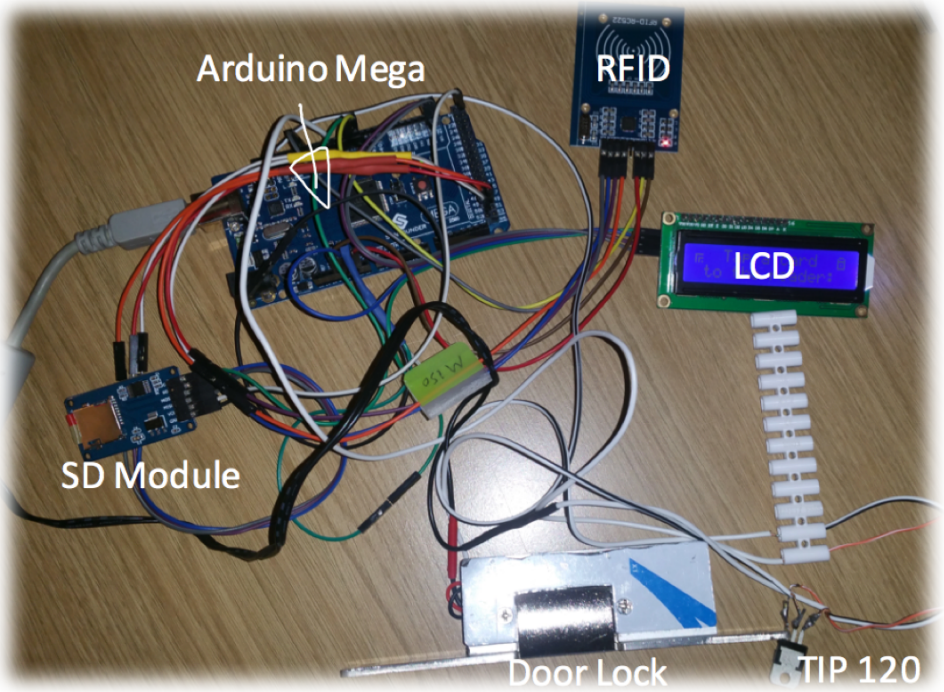
Door Locking Component

SPECIFIC USER PROTOCOL - ADDING / REMOVING CARDS & DEACTIVATING DOOR LOCK



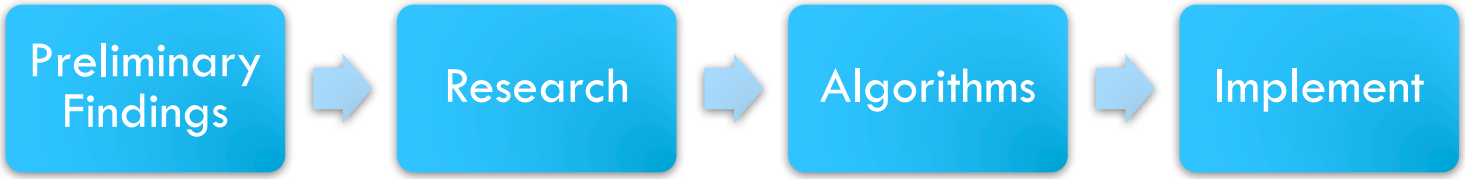
COMPONENTS REQUIRED

Module Name	Description	Objectives achieved by module
SD Module	A device used to read a memory card and input the data into a main controller.	Card Storage
LCD Crystal Display	A 16x2 character display capable of displaying up to 32 characters and even custom ones.	User Interface
RFID Card Reader / Writer	A device capable of reading wireless / RFID based cards. This particular module can also re-writing them.	Card Recognition
Arduino Micro-Controller	A device capable of controlling pre-programmed digital inputs and outputs.	Simultaneous use of all objectives
Electro Magnetic Door Lock	A magnet based system which locks and unlocks a door at the request of the main micro controller.	Door Locking Component

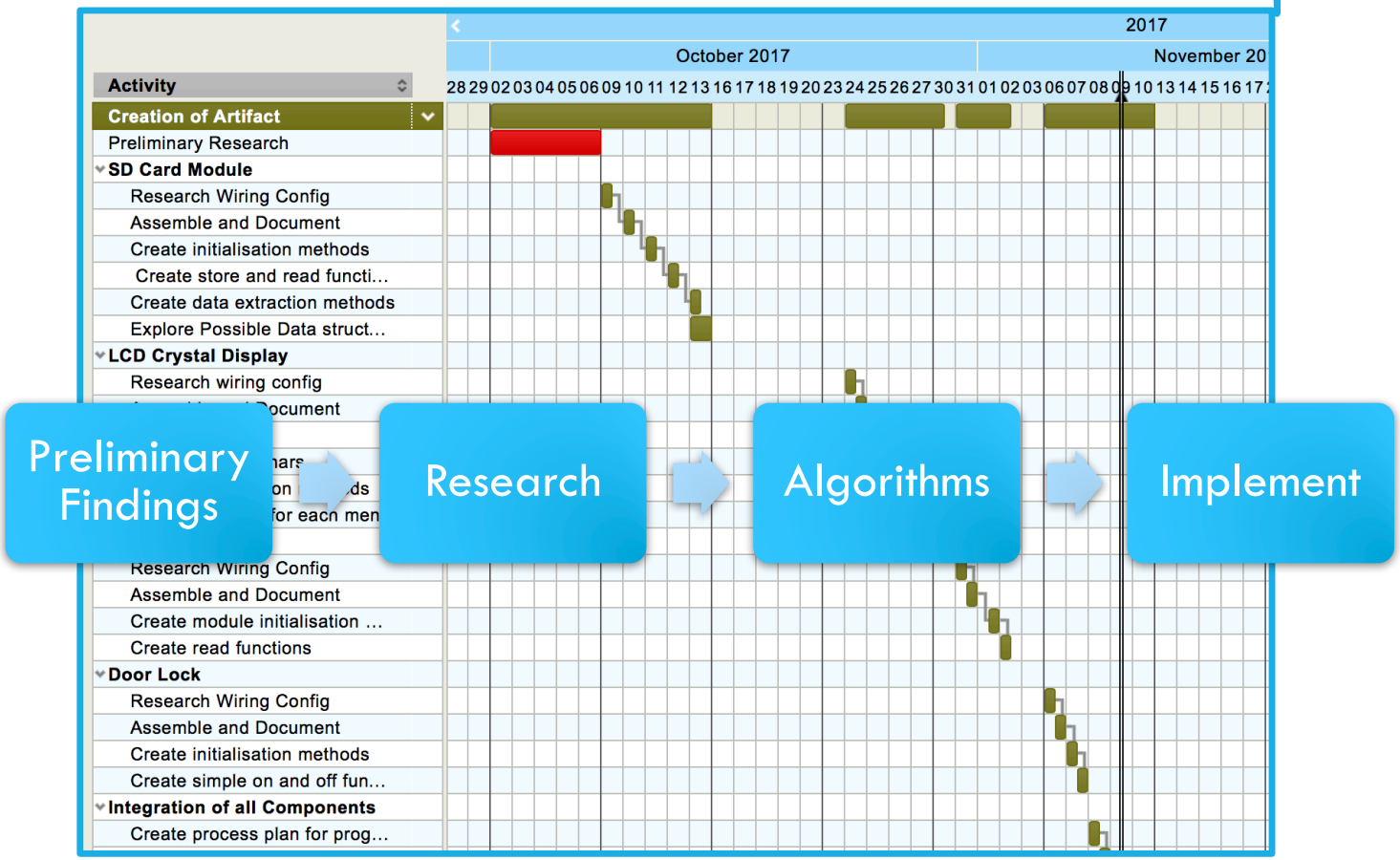


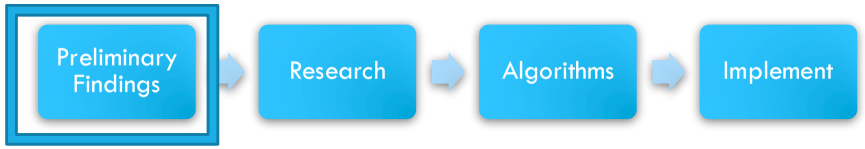
MY RESEARCH AND IMPLEMENTATION PROCESS

This investigation explored the core components of an RFID door lock that are crucial to its operation and how various protocols are applied to it components in order for them to communicate together to create one concurrent operation.



ADOPTING THE PROCESS

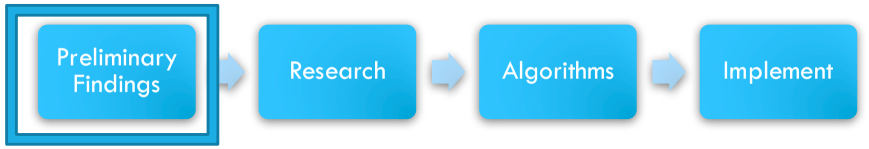




PRELIMINARY FINDINGS

1. Researching the specific pin layout for the module.
2. Sourcing the code libraries required.
3. Implementing the module on its own (to learn the protocol)

ℹ Pin Layout: a reference to the pins / contacts that connect an electrical device.

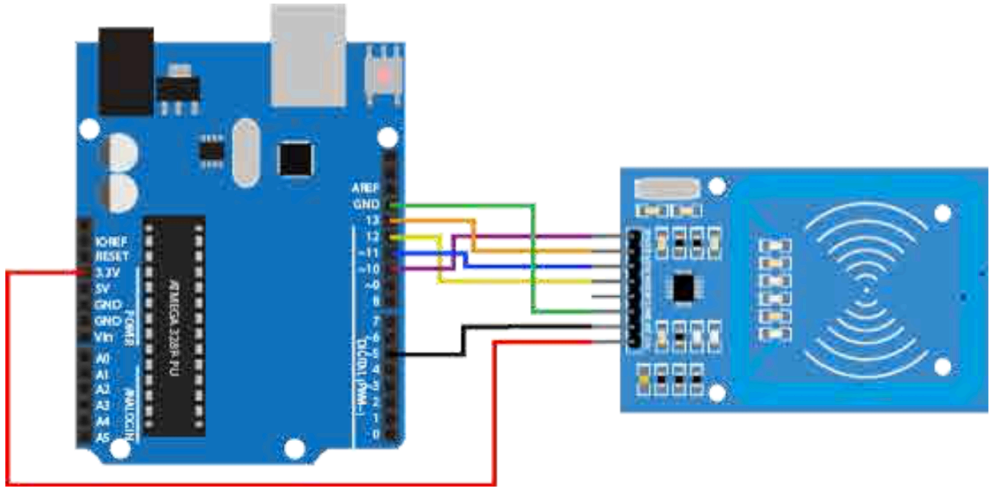


1 - PIN LAYOUT

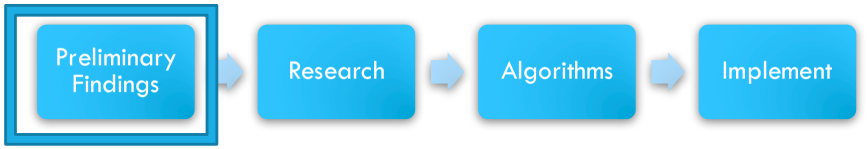
Sources from online documentation.

My preliminary findings showed that the there were pin conflicts between modules, this would later be resolved in my research stage.

RFID-RC522 Module	Arduino Uno
1 - SDA	Digital 10
2 - SCK	Digital 13
3 - MOSI	Digital 11
4 - MISO	Digital 12
5 - IRQ	--unconnected--
6 - GND	Gnd
7 - RST	Digital 5
8 - 3.3V	3.3v

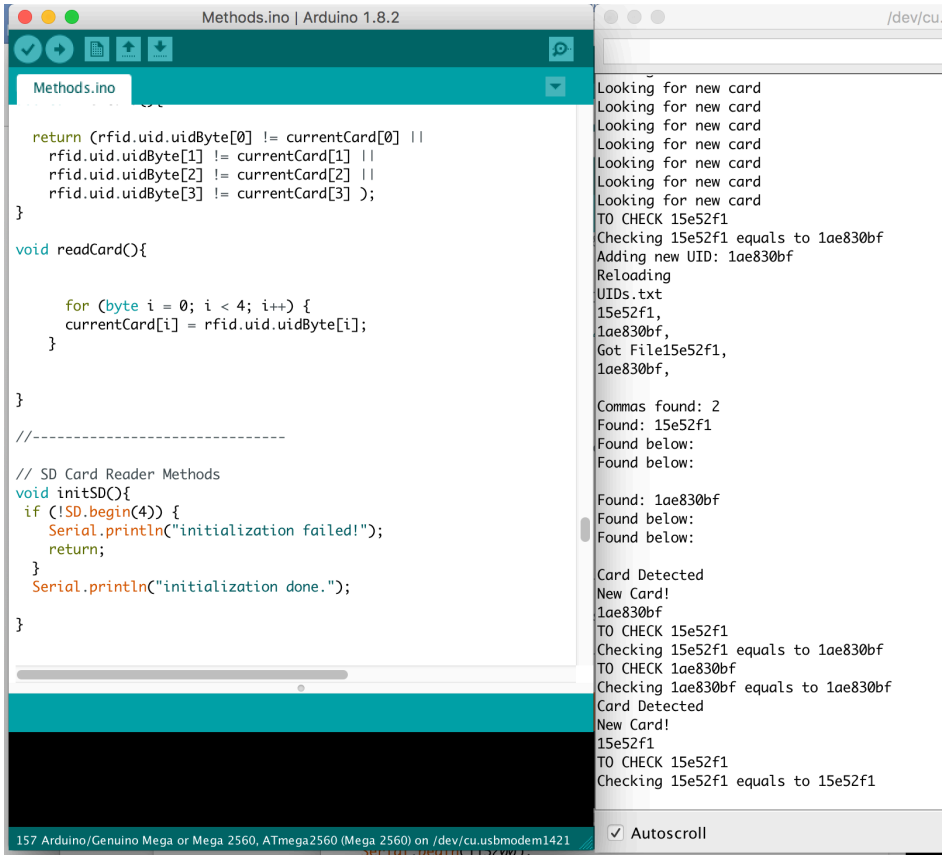


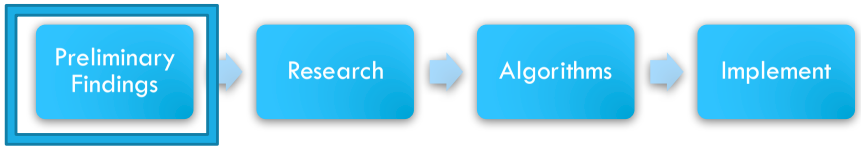
Source: https://cdn.instructables.com/ORIG/FF3/VM8X/J282W5KG/FF3VM8XJ282W5KG.pdf



2 & 3- THE PROGRAMMING & IMPLEMENTATION

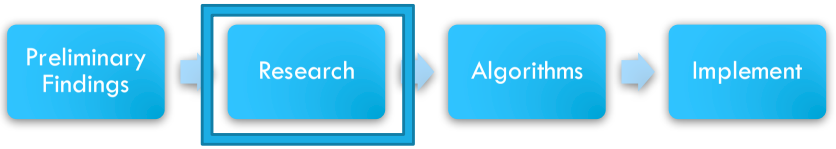
The project was written in C/ C++ using the Arduino IDE





THE RESEARCH

1. Exploring sub components and there efficiencies
2. Exploring the protocols in which the communicate with the main board
3. Using research to solve problems



1- EFFICIENCIES

For LCD screen for example:

Usually requires 16 pins, with the IC2, only 5 pins

IC2 acts as a slave device, receiving and displaying messages using a serial peripheral interface

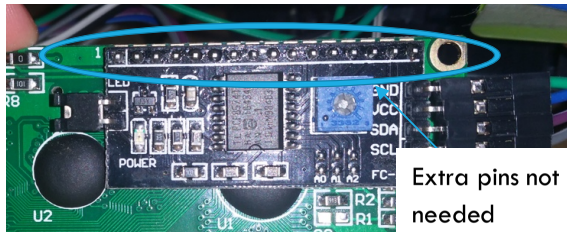
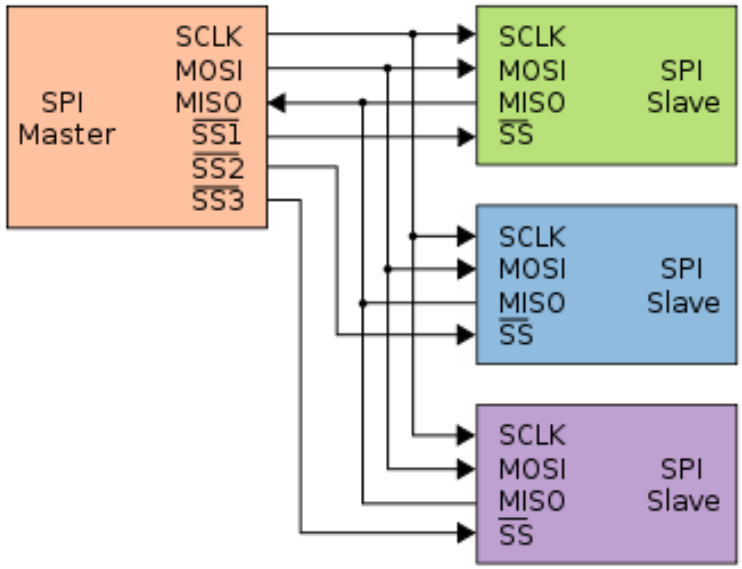
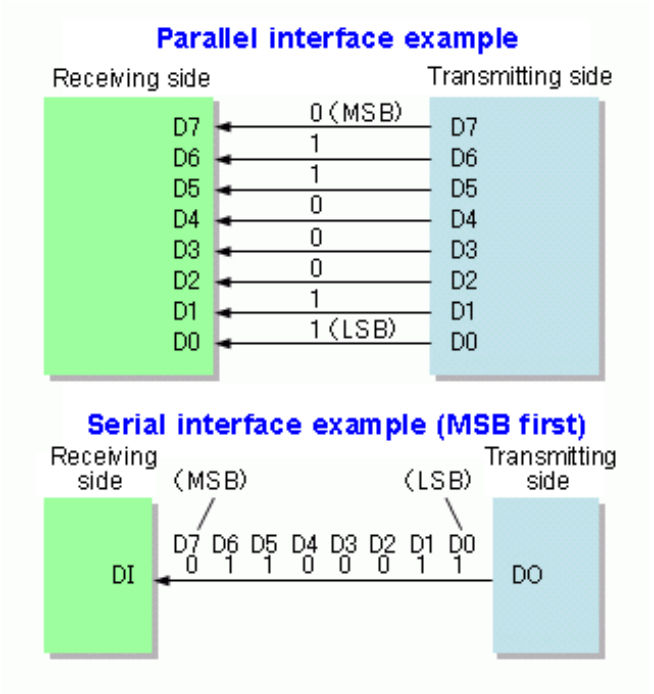


Figure 2c (Evidence)

2- CONCURRENT COMMUNICATION + EXAMPLE

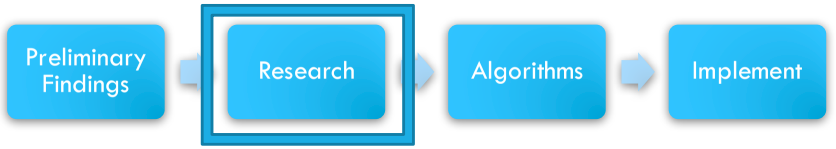
The modules utilize a communication protocol called Serial Peripheral interface (SPI):





Data is passed in (Slave -> Master) using the MISO serial pin, and then passed back out (Master -> Slave) using the MOSI serial pin.

ℹ Concurrency: Executing units of a program in order without effecting the flow or speed.



3 – SOLVING PROBLEMS

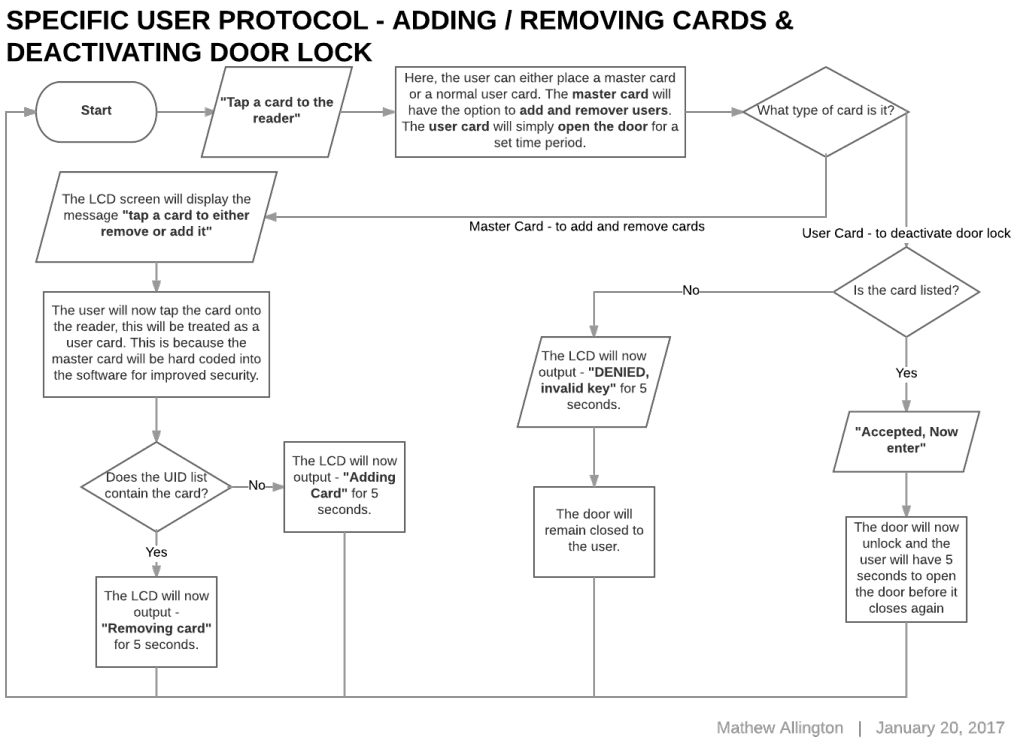
The conflict of SD and RFID pins with the MOSI and MISO pins were resolved.

I did however notice that the reset pins had to be separate:

```
#define RST_PIN 3 // Changed to 3 because of conflict with SD module
```

ℹ MOSI and MISO: Pins used for communication between modules.

MODELLING THE ALGORITHMS

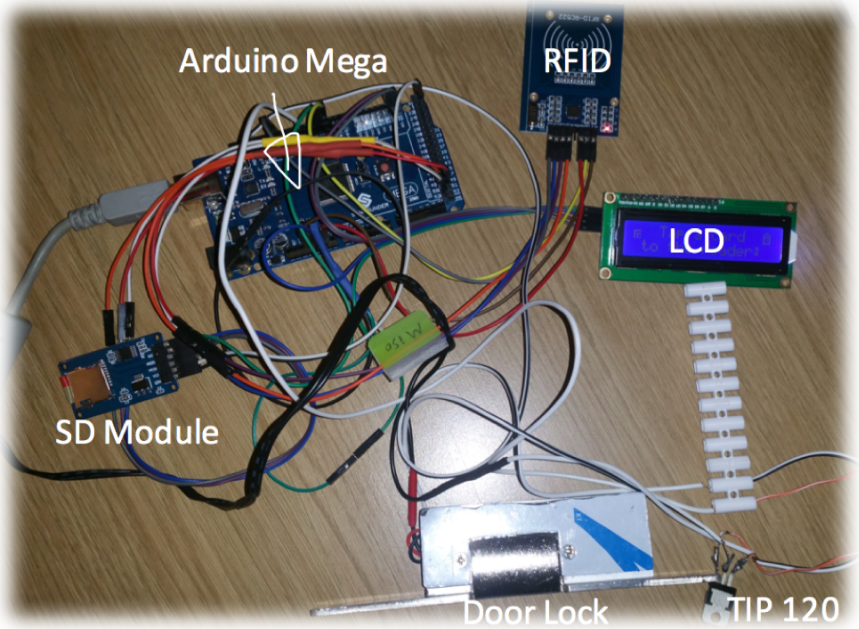



```
FUNCTION setup(self):
    setUpLock()
    setLockOn(True)
    Serial.begin(9600)
    OUTPUT "Lock on"
    setUpLCD("", "")
    OUTPUT "Lcd on"
    splashScreen()
    OUTPUT "Splash Screen up"
    setUpReader()
    initSD()
    storedUIDs <- loadUIDs()
    printArray(storedUIDs, UIDsAmount)
    OUTPUT String("Gathered") += String(UIDsAmount)
+= String(" UIDs")
    homeScreen()
  ENDFUNCTION
```

IMPLEMENTATION

From the research and preliminary findings, I was able to create a final table of the inputs and outputs:

Arduino to all modules (Figure 5a)	
Arduino Mega	**Modules**
5V	SD, LCD
3.3V	RFID
GND	SD, RFID, LCD, Door Lock
D (Digital) 10 - SDA	RFID
D7	Door Lock
D4	SD Reset
D3	RFID Reset
D20 – SDA2	LCD
D21 - SCL	LCD
D50 - MISO	SD, RFID
D51 - MOSI	SD, RFID
D52 - SCK	SD, RFID



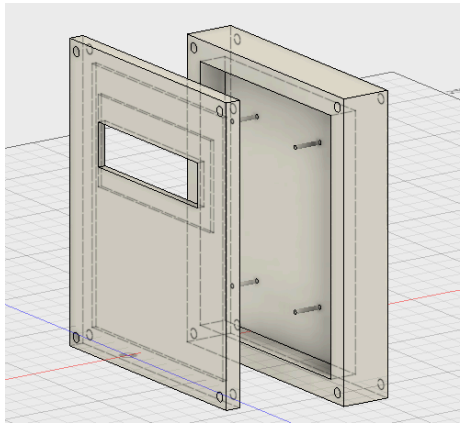
PROBLEMS SQUASHED BY RESEARCH

1. SD Card Initialisation
2. LCD Display – random characters
3. Arduino – Limited flash and dynamic memory

MEASURING SUCCESS

- ✓ To successfully create a system in which I can add remove users
- ✓ Store UID numbers on an SD module or internal Arduino
- ✓ Get the RFID module to read UIDs
- ✓ Get the magnetic door lock to open and close

THE RESULTANT PRODUCT



LESSONS LEARNT

Accountability for potential problems arising

Research contributes a great amount to the success of overcoming potential problems

Proper planning helps structure the project in an orderly manner

THE POSSIBILITIES

With new knowledge acquired, I would like to go ahead and apply it to future systems that I build.

For example,

1. Networked Door Access
2. IFTTT applications
3. Mobile applications
4. Shop floor systems
5. Research of security exploits

THANKS FOR WATCHING