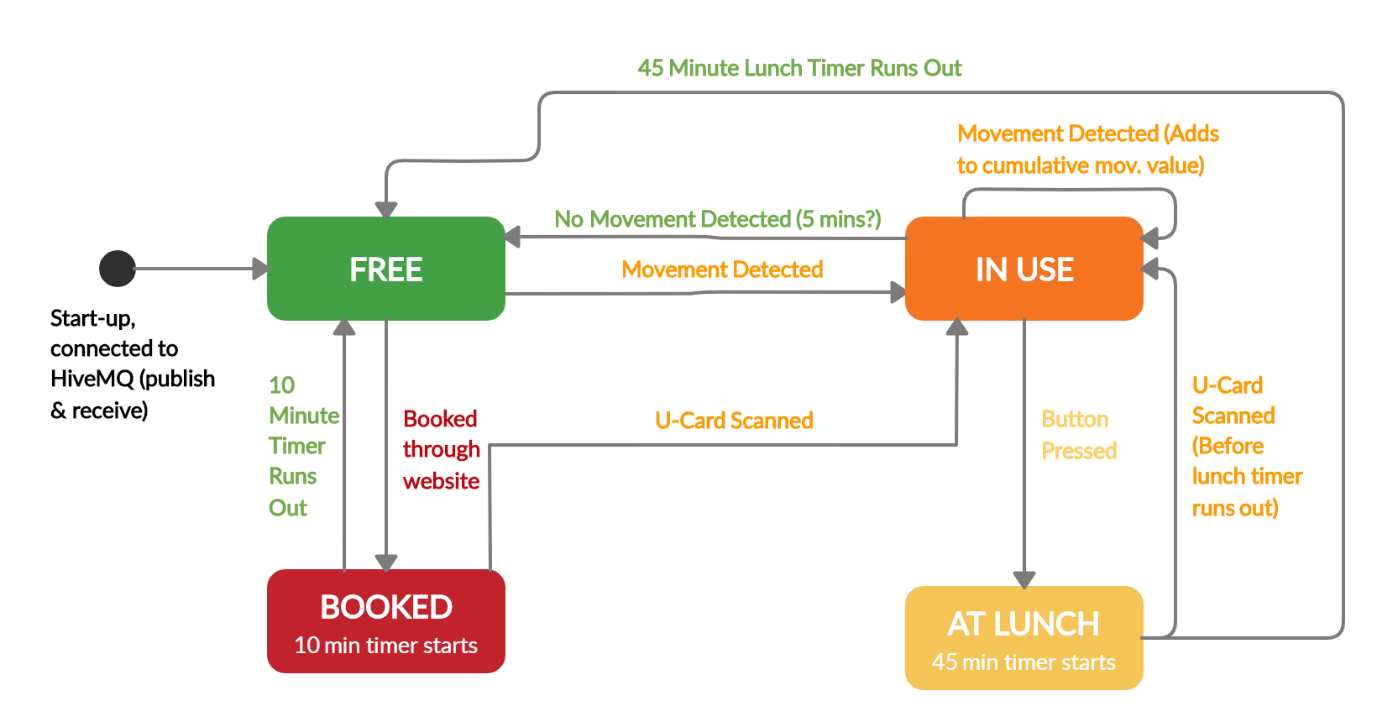
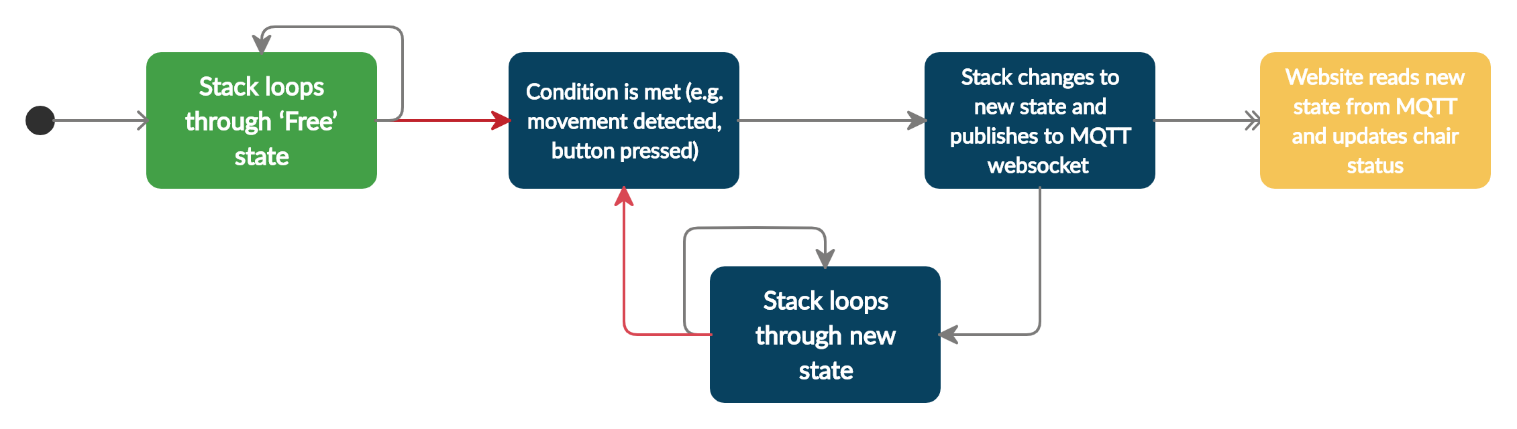
**M5 Stack/Stick Code**

****Both devices are programmed to follow the below state machine:

**Design Process:**

Initially had the stack set to free, then a lot of ‘if’ statements testing for the conditions above all in one loop. Once conditions were met, the appropriate message was published to MQTT (‘booked’, ‘use’, etc) which triggered the callback function when the stack read its own messages. The callback function then printed the appropriate screen to the stack and changed the state.

* This caused a significant delay between conditions changing (i.e. detecting movement) and the screen changing because it had to publish then read back its own message.
* Code was very messy with no clear functions, and trying to set a timer was very difficult.
* The ‘In Use’ mode only changed based on current movement.

Revised code design compartmentalised the states in to functions which run while loops until conditions are met to change them. Code now follows below process:

This refactoring removes the delay from publishing and reading, while still publishing the chair state for the website to update and keep track of.

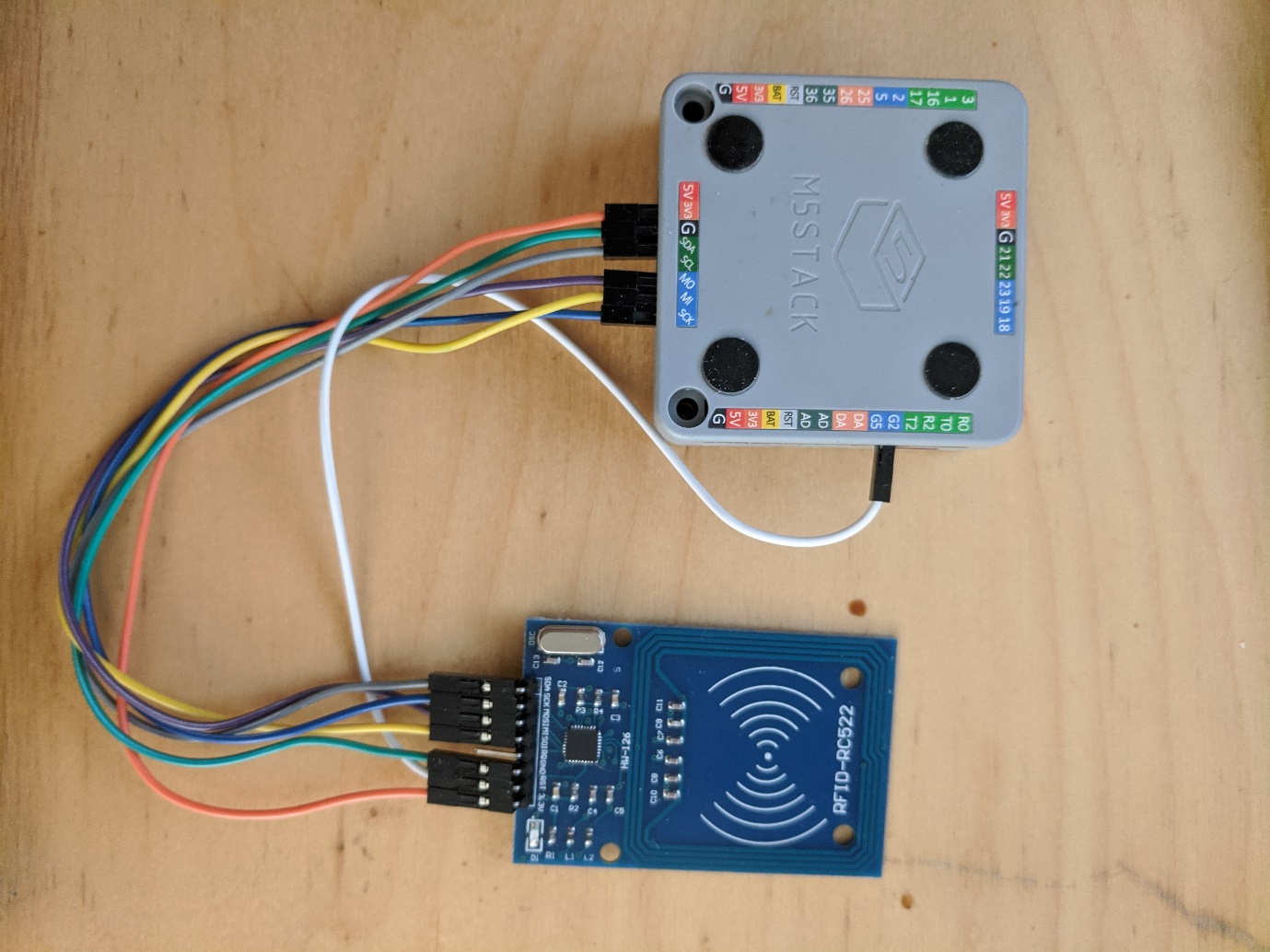
‘Use’ mode now tracks cumulative movement and has an initial starting value which decreases with time and increases when it detects movement (proportional to the amount of movement detected). This prevents the state from reverting back to ‘Free’ as soon as no movement is detected, and also stops the chair from staying in ‘Use’ when it is knocked by somebody walking past.

‘Lunch’ mode on the stick differs from the stack because of the RFID capabilities. Stack reverts to ‘Use’ from ‘Lunch’ when a card is scanned on the stack, but when it detects continued movement on the stick. ‘Booked’ is also disabled by pressing the button on the stick, but scanning a card on the stack.

**Implementing UCard (RFID) scanning functionality:**

In order to confirm the correct individual is using the chair, the student will need to scan their UCard upon arrival. This must be completed within 10 minutes of booking or the chair will be re-released into the pool of available chairs. The UCard acts as an RFID (Radio Frequency Identification Device) and its unique identifier is compared with the ID sent from the web app. If the ID sent from the web app matches the ID scanned, then the student has been successfully authenticated and can begin using the desk.

Reading the ID within the UCard would be achieved using an external RFID reader. The module chosen to achieve this was the MFRC522. The MFRC522 is a highly integrated reader/writer IC for contactless communication at 13.56 MHz. This board is manufactured by NXP but can be purchased from the M5Stack store in an enclosure that connects to the stack’s 4-pin ‘GROVE’ connector. As the delivery times from the online store were large, the standalone MFRC522 board was purchased from eBay, this unit did not include a GROVE interface however. This module was therefore connected to the stack’s I/O bus via jump cables:



The MFRC522 supports two communication protocols, I2C (Inter-Integrated Circuit) and SPO (Serial Peripheral Interface). I2C is sometimes preferred over SPI because it allows for multiple ‘slave’ devices to communicate with multiple ‘masters’ (the stack in this case) using only two lines. One to transmit the data (SDA) and one to carry a synchronising clock signal (SCL). As there are only two lines, I2C requires each slave to be addressed and this is reflected in the MFRC522 object within the source code where the constructor requires an address to be provided when I2C mode is utilised:

*MFRC522(byte chipAddress);*

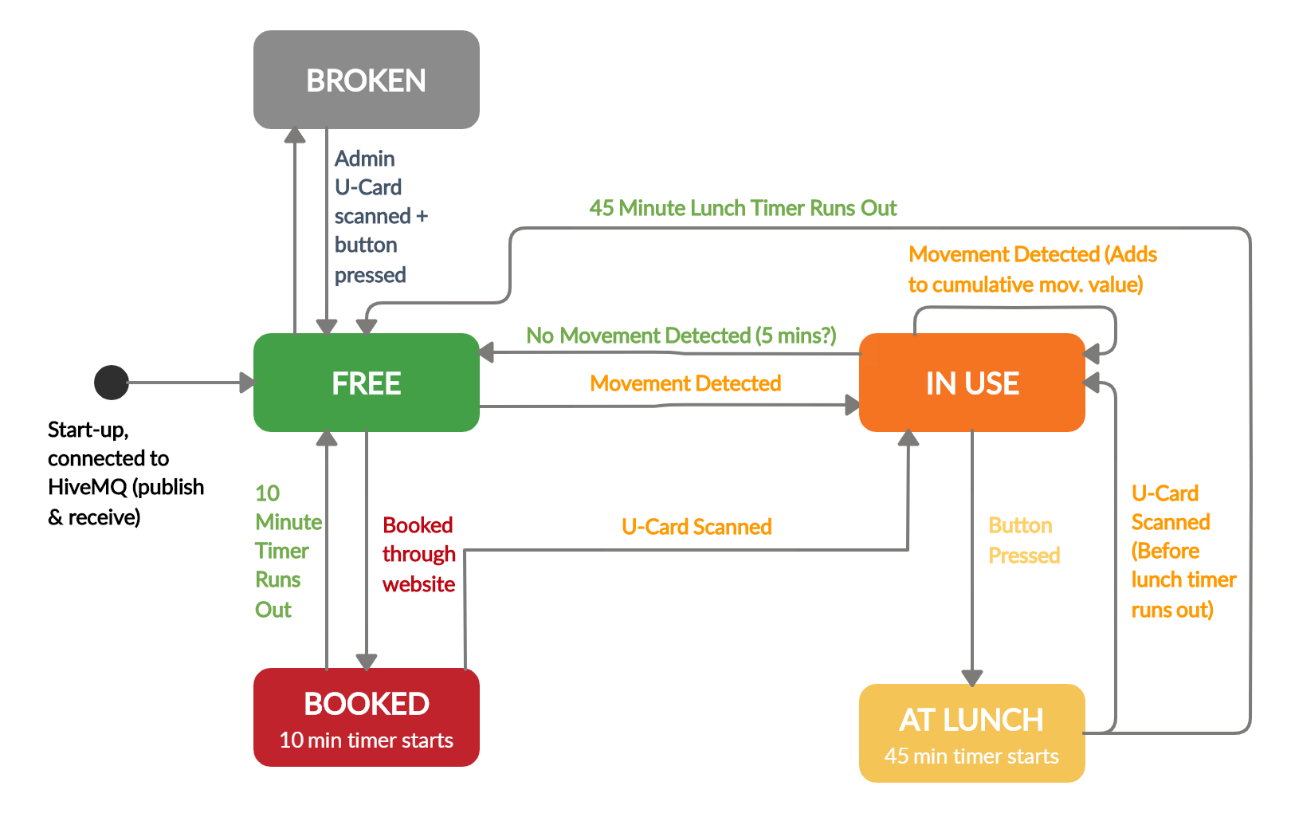
SPI offers faster data rates compared to I2C however needs at least 4 lines to be utilised, a Clock (SCK), a line to transfer data from the CPU to the peripheral MOSI, a line to transfer data from the peripheral back to the CPU (MISO) and a chip select line (CS) to select the peripheral in question (hence many more lines may be needed for multiple devices).

The constructor for the MFRC object in the source code utilising SPI therefore requires a requires the pin value of the peripheral to be selected as a parameter:

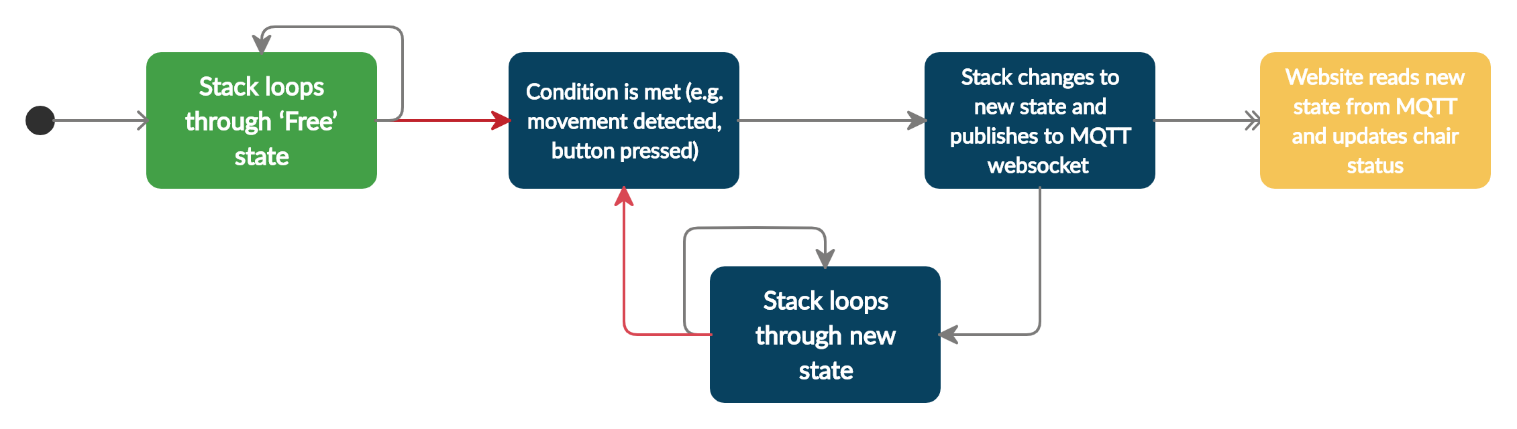
*MFRC522(byte chipSelectPin, byte resetPowerDownPin);*

Unfortunately, we were unable to get the MFRC522 to interface with the stack, this using either of these connection protocols. We were not sure whether the This may have been due to the fact that eBay bought MFRC522’s can often be temperamental.

**Portfolio Points:**

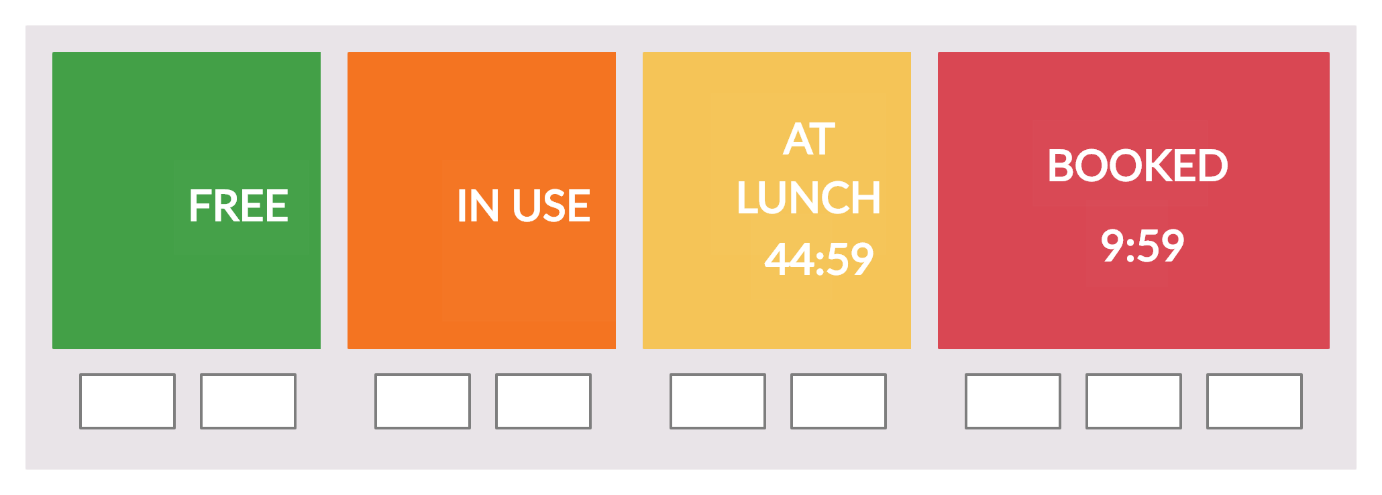
**b) Object-Oriented design of key sub-systems (e.g. Desktop Application, Web Application etc.)**

*M5Stack State Machine Diagram*

The M5 Stack subsystem is responsible for detecting the current state of the chair and displaying it, which is determined by following the above state machine diagram. The stack initialises at the black circle then moves to free, then stays in free until one of the stated conditions is met. It can only then move between the states which are connected by the arrows shown above (e.g. it cannot move from ‘booked’ to ‘at lunch’ directly). Once it changes between two states, it then publishes it’s new state to the MQTT websocket (which the web client receives and updates on the website and within the database). This can be seen in the diagram below where a condition is met, the state is changed and published to MQTT, and then the stack loops through its new state until another condition is met and the state changes again.

The stack can also receive information from the website and send information to the processing application. Both of these are compartmentalised and it is only able to receive the ‘booked’ command from the website when it is in ‘free’ mode (as dictated by the state machine), and it will only send out information to the processing application when it enters the ‘broken’ state (this is also achieved through the use of MQTT).

**d) The evolution of UI wireframes for key sub-systems**



*M5Stack Initial UI Designs*