Lab 7

November 2, 2021

Delivery date: November 2th, 2021

SPI and SD Interfacing with Arduino Zero

Javier Mondragon Martin del Campo

A01365137

Prof. Matías Vázquez Piñón
Tecnológico de Monterrey

1 Introduction

1.1 Explain what you did in this laboratory.

During this lab I went through the SPI module inside the Arduino Zero. The protocol was used to communicate between an SD card shield and to produce a signal displayed on the oscilloscope. For this, the microcontroller was configured with the spi instructions given by the data sheet and instructions on the github provided by the teacher.

1.2 Include a brief explanation of each .C file written for your project.

There was a code imported from another laboratories, this was the UART configuration codes. The explanation for these files are in previous labs.
The SPI code is in spi.c. The general code was provided by the teacher github. The constants and registers were obtained from the datasheet in order to communicate with the SD Shield and generate the signal for the oscilloscope.
The main file has general functions and the request for the SPI and UART functions.

2 Results

2.1 Part I

Flowchart illustrating the initializing process for the SPI:

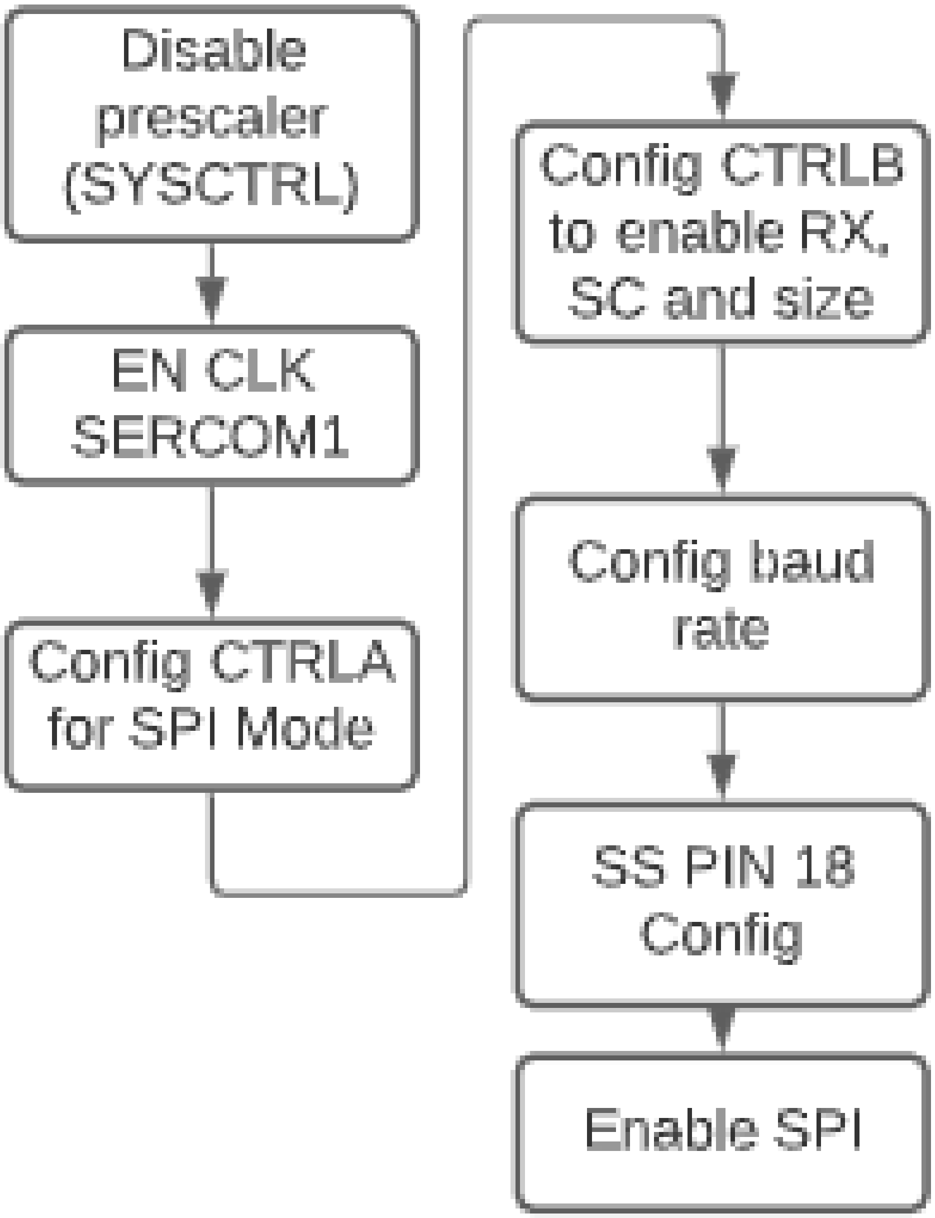

Disable prescaler (SYSCTRL)
EN CLK SERCOM1
Config CTRLA for SPI Mode
Config CTRLB to enable RX, SC and size
Config baud rate
SS PIN 18 Config
Enable SPI

Figure 1: Initializing process for the SPI

Flowchart illustrating the process to send values using spiSend() in spi.c:

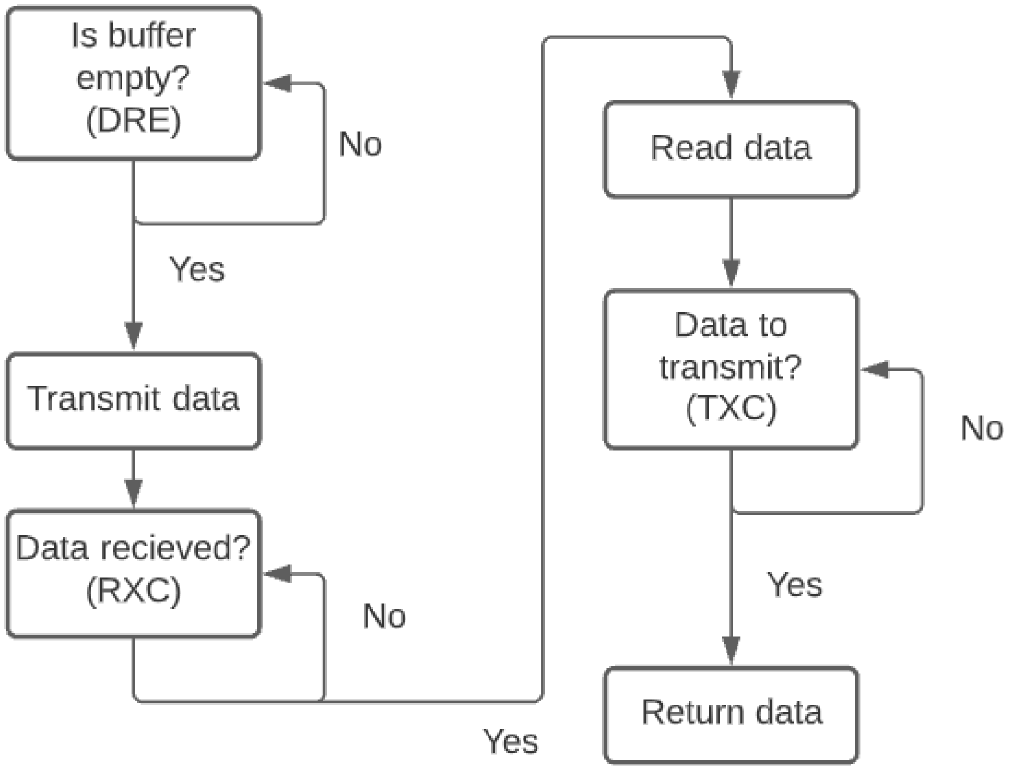


Figure 2: Send values using spiSend

Image of the generated SPI signal over the MOSI and SS lines, including justification of the observed waveform. Was the signal observed on the oscilloscope the expected waveform?
The intention was to generate a square-pulse signal and the results were as expected with a frequency of around 12.45kHz:

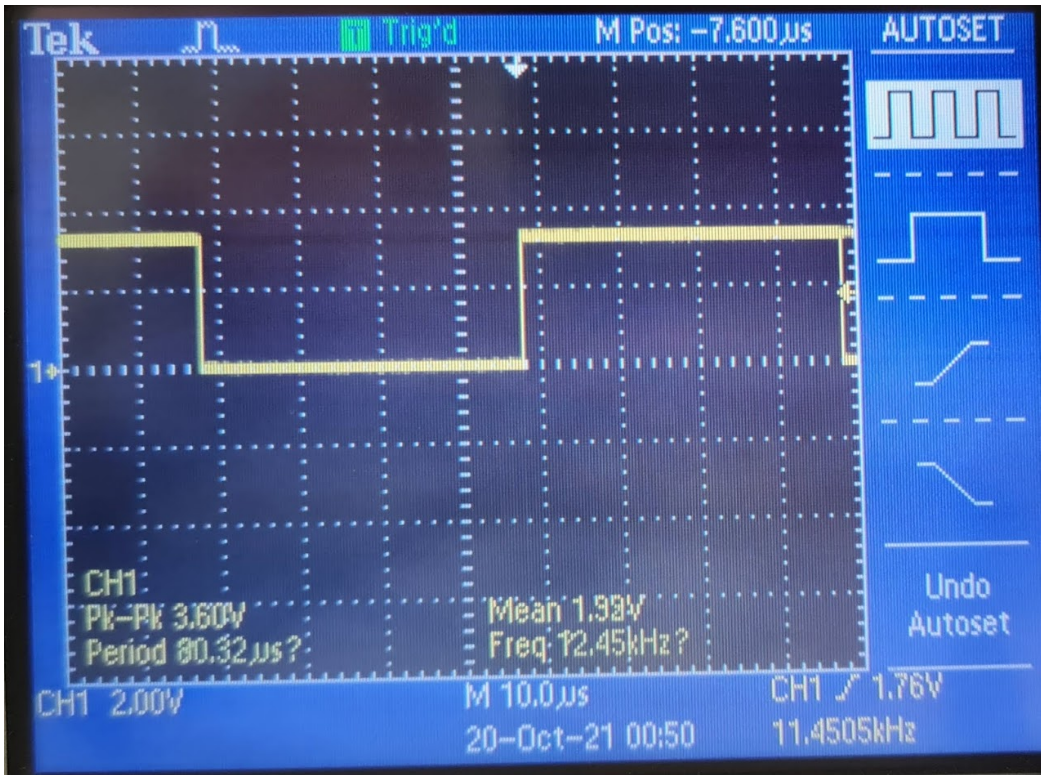


Figure 3: Generated SPI signal

2.2 Part II

Flowchart of the process to send commands to the SD card in the spiXchg() function:

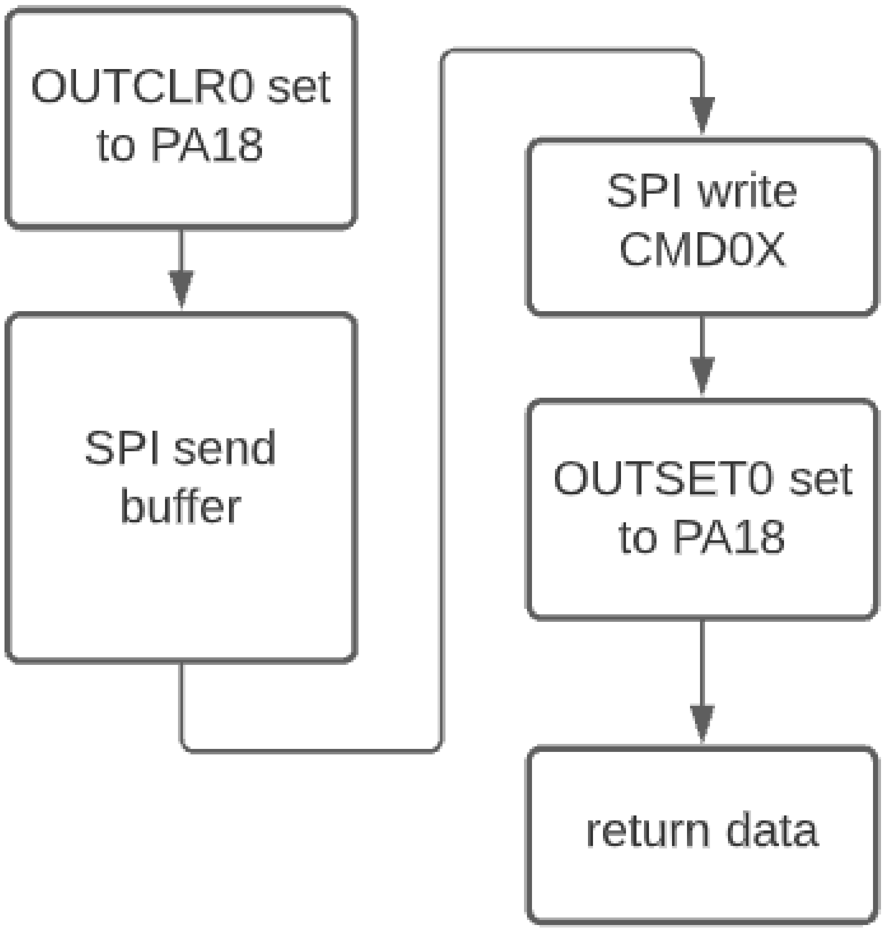


Figure 4: Send commands in the spiXchg()

Flowchart of the process to receive responses from the SD card in the spiXchg() function

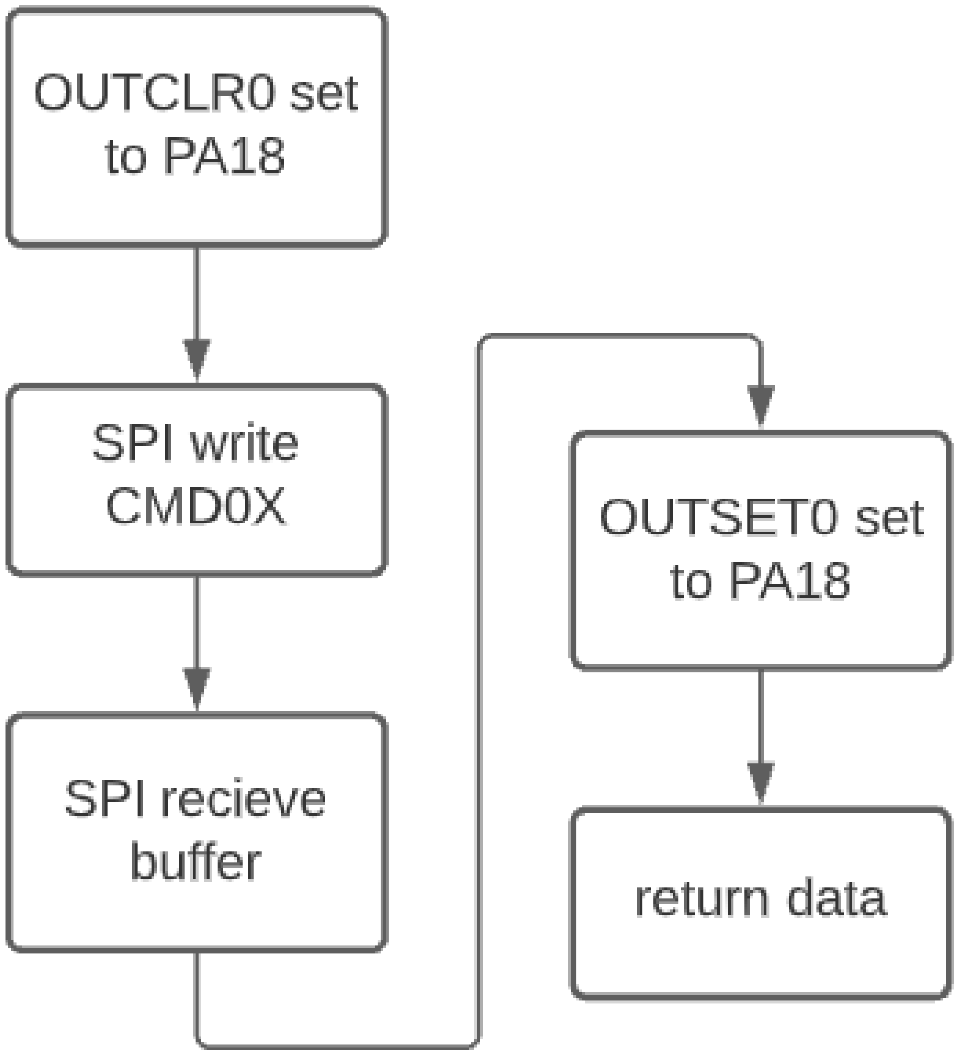
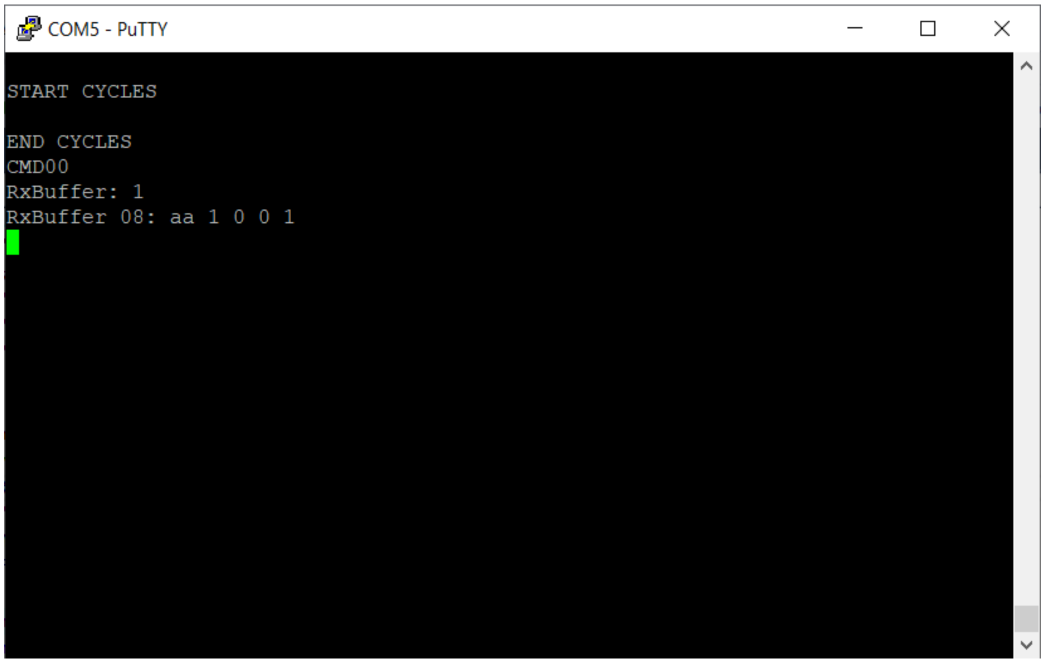


Figure 5: Receive commands in the spiXchg()

Interpretation of responses received from the SD card by sending commands CMD00 and CMD08.
The first response gave me the version and the response that it was a good connection with the SD. The second response gave me the message that the connection was successful.



```
START CYCLES

END CYCLES
CMD00
RxBuffer: 1
RxBuffer 08: aa 1 0 0 1
```

Figure 6: Responses received from the SD card

Answers to questions: When sending each byte of the command, what is the value being received from the SD?
0x01
When receiving the response from the SD, what is the value being sent to the SD?
0xFF

2.3 Part III

What is the purpose of the initCycles() function?
To set the clock cycle for 76

Modifications to the main program to additionally send commands CMD55 and CMD41. Added to the CMD08 the following data: 0x48, 0x00, 0x00, 0x01, 0xAA, 0x87

2.4 Part IV

Flowchart of the process to send commands to the SD card in function rcvr_datablock().

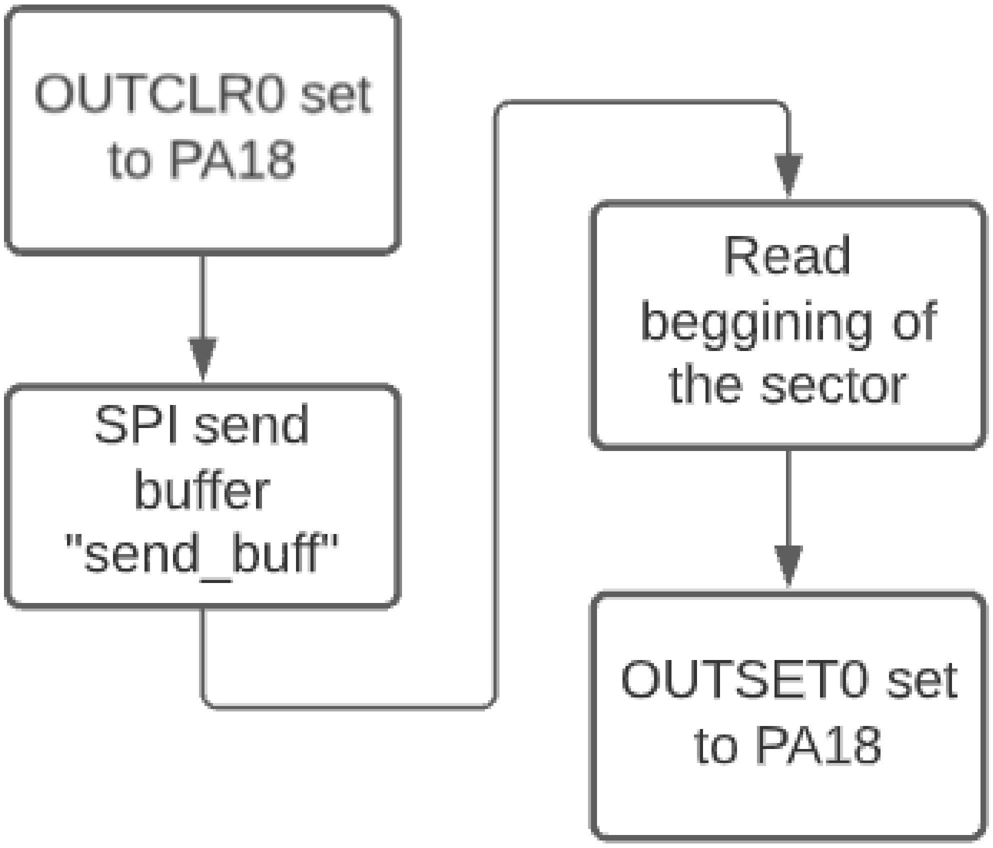


Figure 7: Send commands in function rcvr_datablock()

Answer to the following questions: What arguments of the function are related to this part of the code? What information are these arguments providing?
The first three arguments. Those provide the instruction, a send buffer and a size.
Explain how the CMD17 command argument is being passed to the SD. What is this argument for?
Exactly the same as the CMD08, the argument is to read the data inside a block.
Flowchart of the process to receive response from the SD card in function rcvr_datablock().

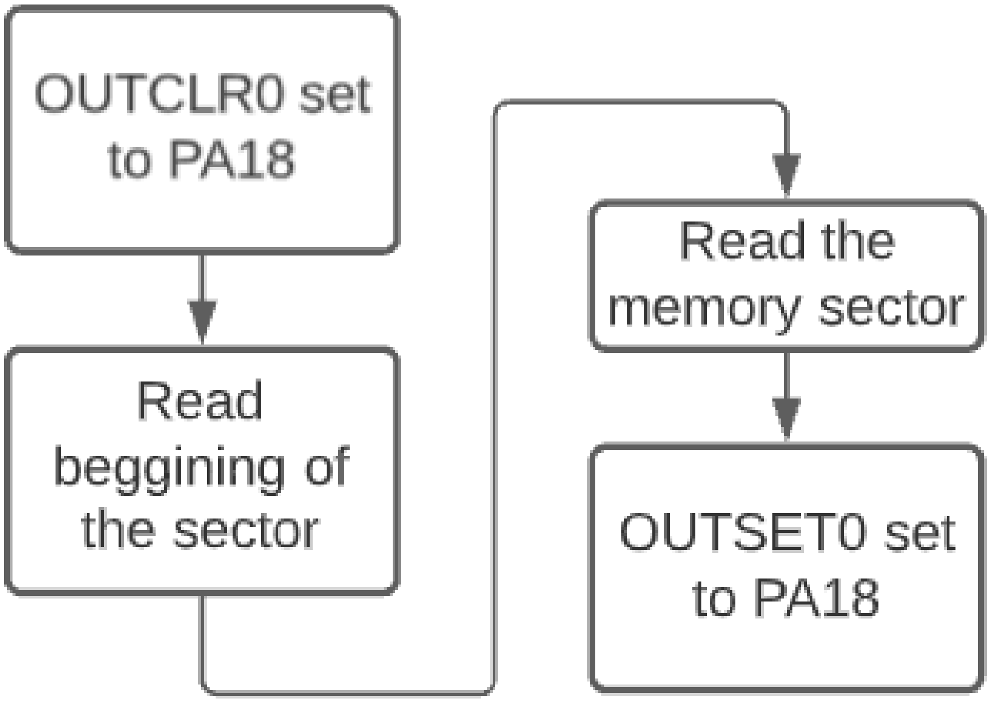
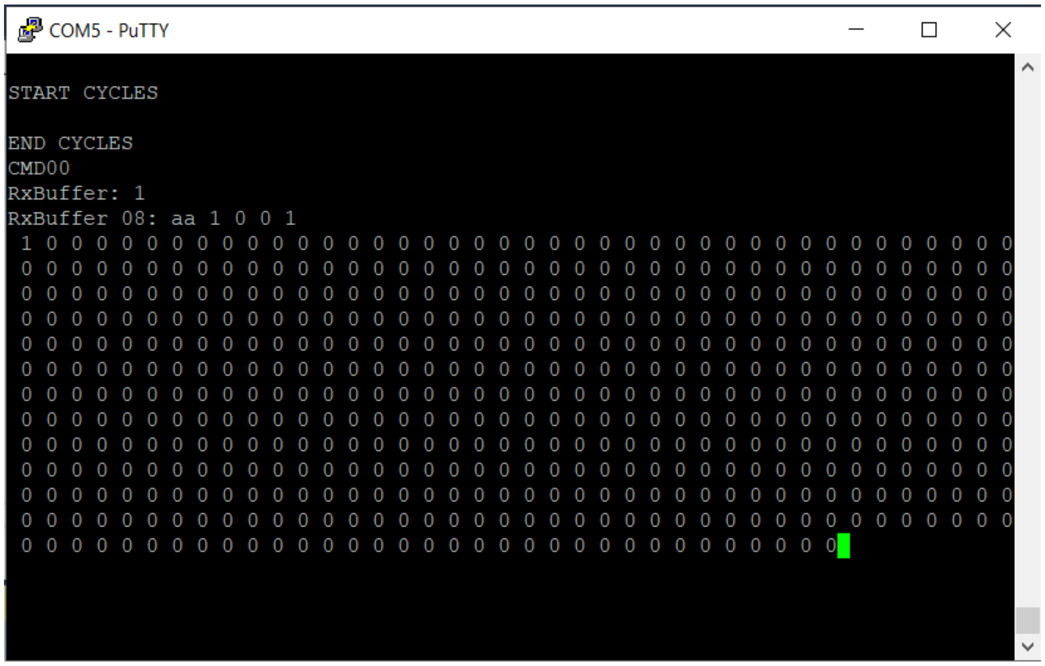


Figure 8: Receive commands in function rcvr_datablock()

What arguments of the rcvr_datablock() function are related to this part of the code?
The last 2 arguments and the first one
What information are these arguments providing?
Type of instruction the SD might do and the buffer with it’s size.
Interpretation of the response received from the SD card by reading the first 512-byte block of the card. The SD was recently formated, that is why the first block appears as entirely 0’s

Content of the memory locations of the first 512-byte block of the SD card.



```
START CYCLES

END CYCLES
CMD00
RxBuffer: 1
RxBuffer 08: aa 1 0 0 1
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 9: Responses received from the SD card

3 Individual conclusions

During this lab there was tricky to get the connections right because the model of my SD shield was not so common. Besides that I figure out how to implement the SPI protocol and combine it along side an UART communication. These gave me tools to learn and explore new forms of communication protocols that I could use in embedded systems.

4 Appendix A

`https://github.com/javiermomc/Sistemas_Embebidos/tree/main/L07`

5 Bibliography

1. `https://github.com/matias-vazquez/SistemasEmbebidos`

2. `http://www.dejazzer.com/ee379/lecture_notes/lec12_sd_card.pdf`