Univeristà degli Studi di Palermo

Embedded Systems

Door Alarm System on Raspberry Pi 4

Angelo Barbera 2023

Table Of Contents

1 Introduction **1**

2 Hardware **1**
2.1 Raspberry PI 4 . . . 1
2.2 FT232-AZ USB to TTL serial UART adapter . . . 2
2.3 KY-003 Hall sensor . . . 3
2.4 KY-012 Buzzer . . . 3
2.5 LEDs . . . 3
2.6 Push Button . . . 4
2.7 Resistors . . . 4
2.8 LCD 1602 . . . 4
2.9 PCF8574AT 8-bit I/O expander for I2C bus . . . 7
2.9.1 The I2C bus . . . 7
2.9.2 LCD connection . . . 9
2.10 GPIO wiring diagram . . . 10

3 Environment **11**
3.1 pijForthos . . . 11
3.2 Minicom, Picocom . . . 11

4 Software **12**
4.1 Environment setup . . . 12
4.2 Code structure . . . 12

5 References **12**

1 Introduction

The project is a Door Alarm System. The System is able to monitor the open or closed status of a door using a hall sensor to detect the presence of a magnet; if the latter is far from the sensor, it is emitted a sound alert with a buzzer. Using two LEDs and a LCD display 16x2 the status of the door is shown. In addition there is a button that can be used to turn off the alarm when the door is closed.
In the following sections is described in detail the hardware and the software used to implement the project.

2 Hardware

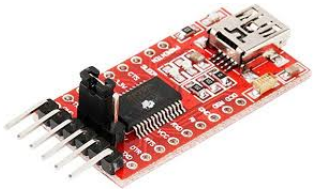
2.1 Raspberry PI 4

The chosen target for this project is the Raspberry Pi 4 Model B, a single board computer developed by the Raspberry Pi Foundation and realeased in 2019. The tech specs include:

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5Ghz
- 1GB, 2GB, 4GB, or 8GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 Ghz 802.11ac wireless
- Gigabit Ethernet
- Bluetooth 5.0, BLE
- 2 USB 3.0 ports, 2 USB 2.0 ports
- Raspberry Pi standard 40 pin GPIO header

- 2 micro-HDMI ports (up to 4kp60 supported)
- 2-lane MIPI DSI display port
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H.264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.1, Vulkan 1.0
- Micro-SD card slot for loading operating system and data storage
- 5V DC via USB-C connector (minimum 3A)
- 5V DC via GPIO header (minimum 3A)
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)
- Operating temperature 0 - 50 °C ambient

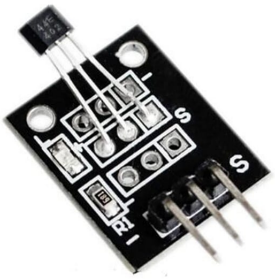
2.2 FT232-AZ USB to TTL serial UART adapter



The FT232-AZ USB to TTL serial UART adapter is used to connect the PC used during the development of the project to the target in order to send and receive data between the PC and the Raspberry Pi 4. The PC is connected through a USB port, the target is connected through GPIO pins according to the following table.

GPIO	Function	UART adapter
14 (Tx)	Output	Rx
15 (Rx)	Input	Tx
Ground	Ground	Ground

2.3 KY-003 Hall sensor



The KY-003 hall sensor allows to detect a magnetic field. When the magnetic field at the Hall sensor exceeds the operate point threshold (BOP) the output of the device switches low. When the magnetic field is reduced to below the realease point threshold (BRP) the device output switches high. BOP and BRP may vary respectively from 1 mT to 33 mT and from 5 mT to 35 mT at operating temperature T = 25° C depending on the sensor model. This sensor is used to trigger the alarm when the magnet is far from the sensor.

2.4 KY-012 Buzzer

The KY-012 Buzzer is an active piezoelectric buzzer, it generates a sound of approximately 2.5kHz when input signal (S) is high. The Buzzer is activated when the Hall sensor does not detect the magnet.

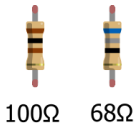
2.5 LEDs

The LEDs are used to show the alarm status. When the Hall sensor does not detects the magnet the green LED turns off and the red LED turns on. When the Hall sensor detect the magnet and the push button is pressed, the green LED turns on and the red LED turns off.

2.6 Push Button

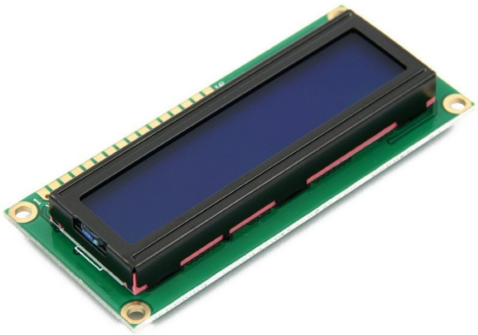
The push button can be used to turn off the alarm when the hall sensor detects the magnet.

2.7 Resistors

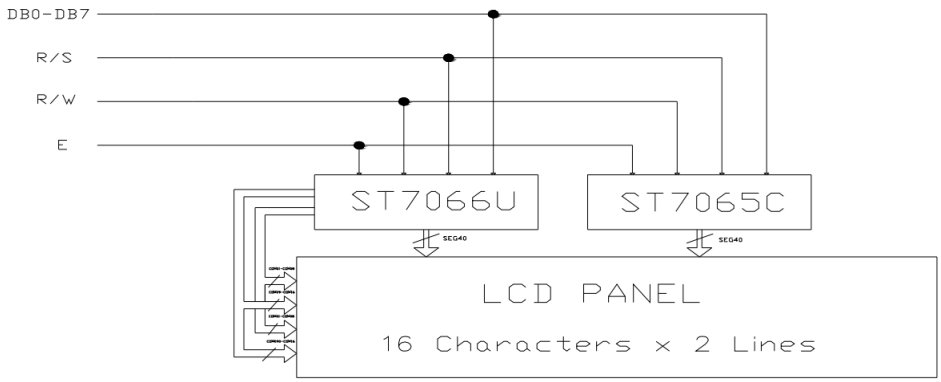


The resistors are connected in series to the LEDs to limit the current flowing through the LED and to ensure that the supplied voltage does not exceeds the maximum voltage of the LED. The 100 Ω resistor is connected to the red LED, the 68 Ω resistor is connected to the green LED.

2.8 LCD 1602



The LCD 1602 is an industrial character LCD that can display 16x02 or 32 characters at the same time. The LCD 1602 is controlled through a parrallel interface with 8-bit / 4-bit data bus and 3 control signals. The interface signals reach the two controller chips that drive the LCD panel as shown in the following block diagram.



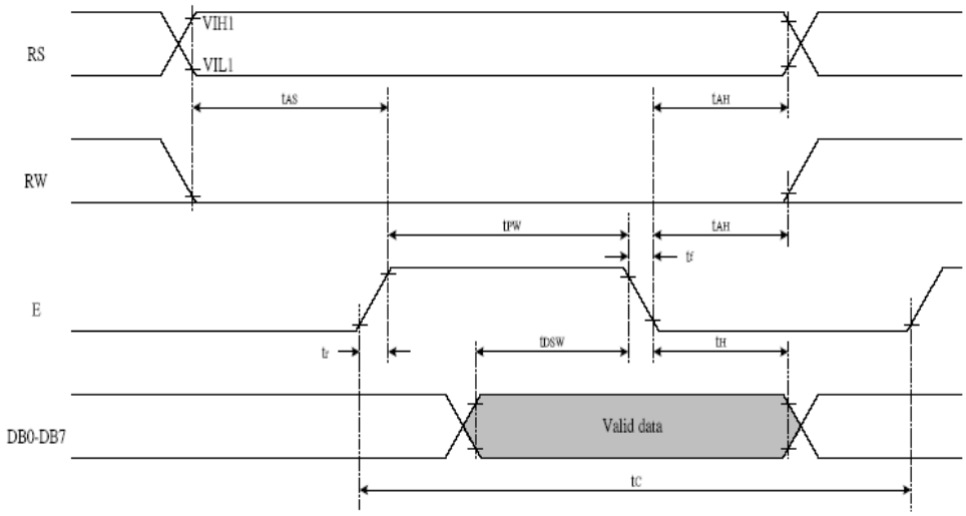
The following table describes the pin assignment

No.	Symbol	Level	Function	
1	Vss	--	0V	Power Supply
2	Vdd	--	+3.3V	
3	V0	--	for LCD	
4	RS	H/L	Register Select: H:Data Input L:Instruction Input	
5	R/W	H/L	H--Read L--Write	
6	E	H,H-L	Enable Signal	
7	DB0	H/L	Data bus used in 8 bit transfer	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L	Data bus for both 4 and 8 bit transfer	
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		
15	BLA	--	BLACKLIGHT +3.3V	
16	BLK	--	BLACKLIGHT 0V-	

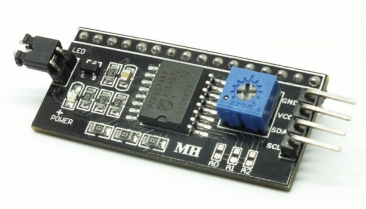
The LCD module is controlled through instructions to set display format, data length, scrolling modality, internal RAM address, to perform data transfer from/to internal RAM and to access status flag. The following table describes the instructions.

Instruction	Instruction Code										Description	Description Time (270KHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM. and set DDRAM address to "00H" from AC	1.52 ms
Return Home	0	0	0	0	0	0	0	0	1	x	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.52 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 us
Display ON/OFF	0	0	0	0	0	0	1	D	C	B	D=1:entire display on C=1:cursor on B=1:cursor position on	37 us
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	x	x	Set cursor moving and display shift control bit, and the direction, without changing DDRAM data.	37 us
Function Set	0	0	0	0	1	DL	N	F	x	x	DL:interface data is 8/4 bits N:number of line is 2/1 F:font size is 5x11/5x8	37 us
Set CGRAM address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter	37 us
Set DDRAM address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	37 us
Read Busy flag and address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0 us
Write data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM)	37 us
Read data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM)	37 us

In order to write to the controller chips we need to set properly the control bits: the R/W bit must be set to 0 for writing, the RS bit must be set to 1 for data input and to 0 for instruction input, the E bit must be set to 1 before the start of the data transmission and must be set to 0 before the end of the data transmission as shown in the following figure



2.9 PCF8574AT 8-bit I/O expander for I2C bus



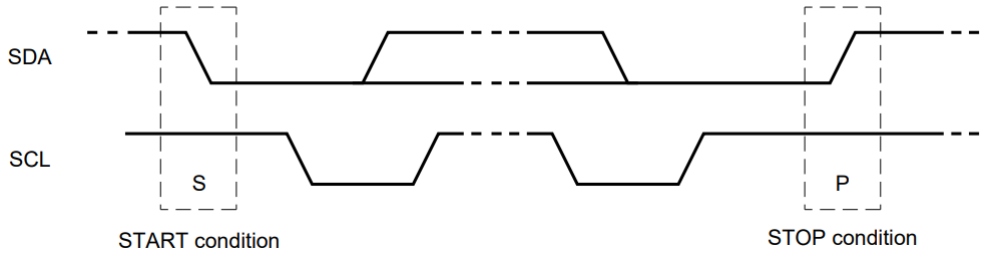
The PCF8574AT provides general-purpose remote I/O expansione via the two-wire bidirectional I^2C bus. It is used to connect the Raspberry Pi to the LCD 1602 using the I^2C bus instead of the parrallel interface.

2.9.1 The I2C bus

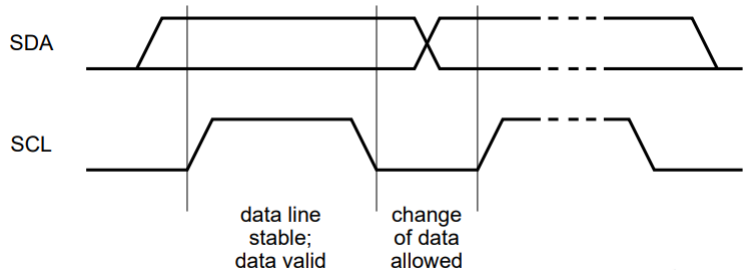
The I^2C bus is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial computer bus invented by Philips Semiconductor. It is used to connect lower speed peripheral integrated circuits to processors and microcontrollers in short distance. Each device connected to the bus is software addressable by a unique address to differentiate between other devices that are on the same bus. Two wires carry data (SDA) and clock signal (SCL).
Open drain connection allows for bidirectional communication: to transmit low the logic activate the pull-down FET (so the line is pulled low), to transmit high the logic realease the bus turning off the pull-down FET (so the pull-up resistor pulls up the line).
The procedure for a master to send data to a slave is the following:

- Master-transmitter sends a START condition and addresses the slave-receiver
- Master-transmitter sends data to slave-receiver
- Master-transmitter terminates the transfer with a STOP condition

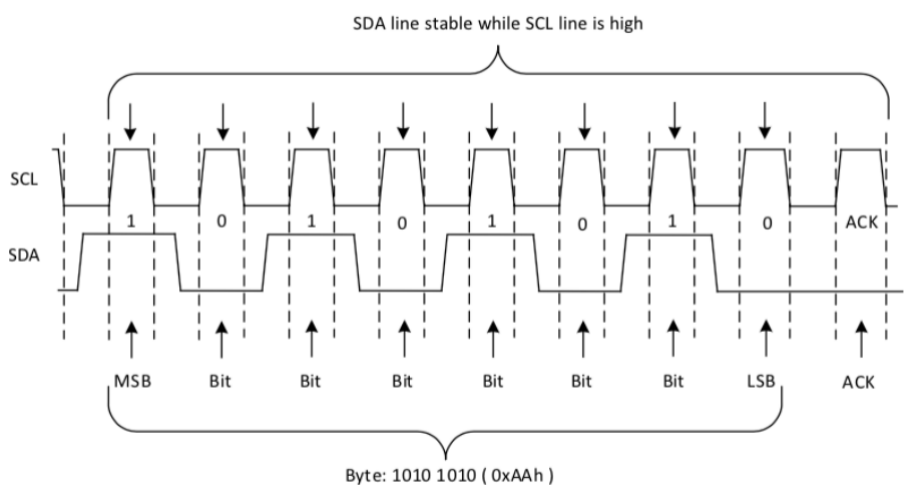
The START condition is defined by a high-to-low transition on the SDA line while the SCL is high. The STOP condition is defined by a low-to-high transition on the SDA line while the SCL is high.



One data bit is transferred during each clock pulse of the SCL, the data on the SDA line must remain stable duriong the high period of the clock pulse as changes in the data line at this time will be interpreted as control signals (START or STOP). Any number of data bytes can be transferred between ther START and STOP conditions. Data is transferred Most Significant Bit first.

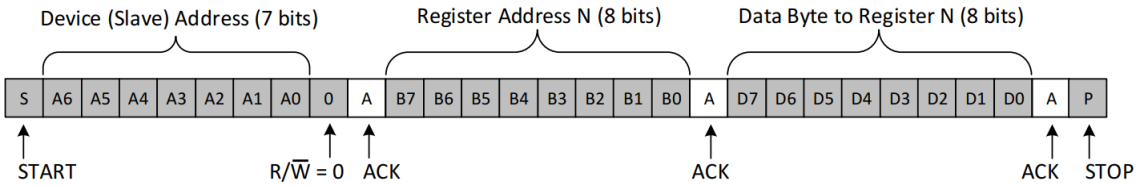


Each byte of data is followed by one ACK (acknowledge) bit from the receiver to signal thath the byte was successfully received. Before the receiver can send an ACK de transmitter must realease the SDA line. To send an ACK bit the receiver pulls down the SDA after receiving the last bit so the line is stable low during the high fase of the ACK clock period. When SDA line remains high after receiving the last bit, this is interpreted as a NACK (not acknowledge)



The procedure to write on the bus is the following:

- The master sends a START condition with the slave's address followed by the R/W bit set to 0
- The slave sends the acknowledge bit
- The master sends the register address of the register it whishes to write to
- The slave acknowledges the register address
- The master starts sending the register data to the slave
- The master terminates the transmission with a STOP condition



2.9.2 LCD connection

The following table describes the connections between the LCD 1602 and the PCF8574AT

PCF8574AT	LCD 1602
P0	RS
P1	RW
P2	E
P3	Backlight
P4	D4
P5	D5
P6	D6
P7	D7

In order to set the LCD 1602 to 4 bit mode it is necessary to send the following command

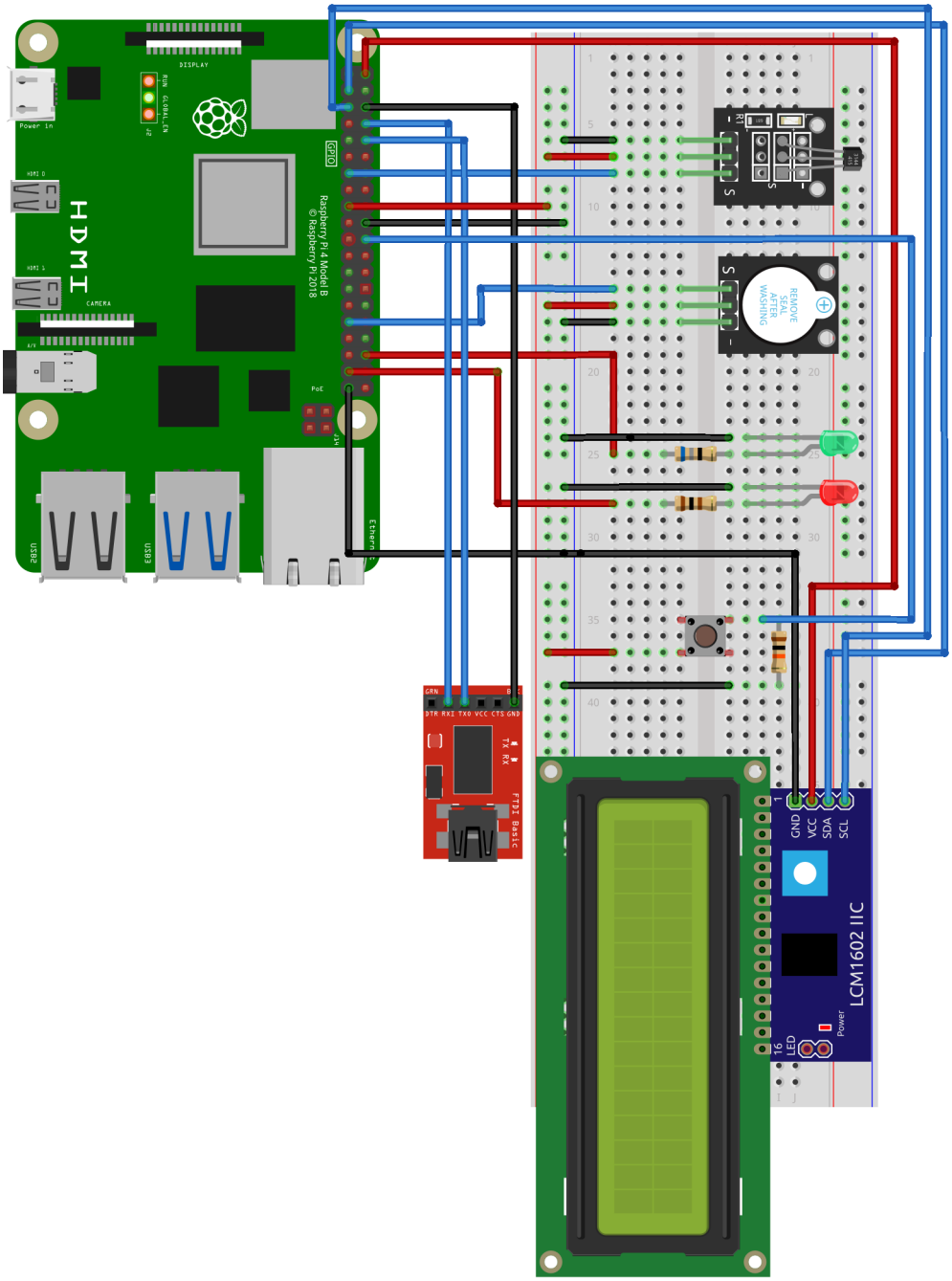
	D7	D6	D5	D4	Backlight	E	R/W	RS
2C	0	0	1	0	1	1	0	0
28	0	0	1	0	1	0	0	0

Now is possible to send a command signal or a data signal according to the following tables

	D7	D6	D5	D4	Backlight	E	R/W	RS
MSB_CMD C	B7	B6	B5	B4	1	1	0	0
MSB_CMD 8	B7	B6	B5	B4	1	0	0	0
LSB_CMD C	B3	B2	B1	B0	1	1	0	0
LSB_CMD 8	B3	B2	B1	B0	1	0	0	0

	D7	D6	D5	D4	Backlight	E	R/W	RS
MSB_DATA D	B7	B6	B5	B4	1	1	0	1
MSB_DATA 9	B7	B6	B5	B4	1	0	0	1
LSB_DATA D	B3	B2	B1	B0	1	1	0	1
LSB_DATA 9	B3	B2	B1	B0	1	0	0	1

2.10 GPIO wiring diagram



The following table is the GPIO wiring diagram.

GPIO	Function	Connection
2	SDA	SDA (I/O expander)
3	SCL	SCL (I/O expander)
6	Output	S (Buzzer)
16	Output	Anode (Green LED)
25	Input	Button
26	Output	Anode (Red LED)
27	Input	S (Hall sensor)
5V	Power	VCC (I/O expander)
3V3	Power	Breadboard
Ground	Ground	Breadboard

3 Environment

3.1 pijForthos

Forth is a procedural, stack-oriented programming lenguage and interactive environment designed by Charles H. "Chuck" Moore. Forth combines a compiler with an integrated command shell, where the user interacts via subroutine called words
pijForthos is a bare-metal operating system for Raspberry Pi based on Jonesforth-ARM. Jonesforth-ARM is an ARM port of x86 JonesForth by Richard W.M. Jones, a Forth interpreter developed for ARM. If you have pijFORTHos on a Micro SD card in the Raspberry Pi, you can connect it to development machine using the USB to TTL serial UART adapter. Using the programs described below you can have access to the Forth console.

3.2 Minicom, Picocom

Minicom is a text-based terminal emulator program for Unix-lile operating systems, it is used to set up a remote serial console. Picocom is similar to Minicom and it was designed to serve as a simple, manual, modem configuration, testing and debugging tool.
ASCII-XFR transfers files in ASCII mode and it is used to send the source file to the Raspberry Pi allowing to set a delay between each character and line sent. It is useful beacuse, since the UART connection is asynchronous, if the reciver is busy compiling or executing Forth words and the transmitter is sending data, it is possible to lose them.
The following command is used to set up the communication with the Raspberry Pi :
`picocom --b 115200 /dev/ttyUSB0 --imap delbs -s "ascii-xfr -sv -l100 -c10"`

- `--b 115200` defines the baud rate
- `/dev/ttyUSB0` specifies the UART port
- `--imap delbs` maps del to backspace
- `-s "ascii-xfr -sv -l100 -c10"` specifies the external program that will be used to transmitting files
- `-sv` specifies verbose mode
- `-l100` specifies the delay (100 milliseconds) after transmitting each line
- `-c10` specifies the delay (10 milliseconds) after transmitting each character

4 Software

4.1 Environment setup

To setup the environment it is necessary to format the micro SD card using FAT32 as file system. Then the following files must be coped in the micro SD card:

- `bootcode.bin` contains the bootloader code
- `config.txt` contains the configuration parameters
- `start4.elf` contains the second-stage bootloader
- `fixup.dat` contains Video Core code and initialization data
- `bcm2711-rpi-4-b.dtb` describes the hadrware present in the Raspberry Pi
- `kernel7.img` contains the pijFORTHos code

The `config.txt` file must be edited specifying the following settings:

- `dtparam=i2c_arm=on` enables I^2C
- `enable_uart=1` enables UART

4.2 Code structure

5 References

- raspberrypi.com/products/raspberry-pi-4-model-b/specifications
- cdn.shopify.com/s/files/1/1509/1638/files/FT232-AZ_Adapter_Datenblatt_AZ-Delivery-_Vertriebs_GmbH.pdf
- cdn.shopify.com/s/files/1/1509/1638/files/Hall_Sensor_Modul_Digital_Datenblatt.pdf
- cdn.shopify.com/s/files/1/1509/1638/files/KY-012_Buzzer_Modul_Aktiv_Datenblatt-_AZ-Delivery_Vertriebs_GmbH.pdf
- wiki.52pi.com/index.php?title=Z-0234
- msb-co.ir/wp-content/uploads/2021/10/eone-1602a1.pdf
- nxp.com/docs/en/data-sheet/PCF8574_PCF8574A.pdf
- ti.com/lit/an/slva704/slva704.pdf
- github.com/organix/pijFORTHos
- linux.die.net/man/1/minicom
- linux.die.net/man/8/picocom
- linux.die.net/man/1/ascii-xfr
- manpages.ubuntu.com/manpages/trusty/man8/picocom.8.html
- unix.com/man-page/redhat/1/ascii-xfr/