NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY POLITEHNICA BUCHAREST
Faculty of Electronics, Telecommunications and Information Technology

MES Project

Stepper motor sequence controller

Author: Savu Constantin-Alexandru ACES

2023-2024

Contents

1 Project description

In the context of this project, a Raspberry Pi Pico has been programmed to control a stepper motor to execute a specific sequence of rotations. Data input is facilitated through the utilization of a 4x4 matrix keypad. The sequence is displayed on an LCD.

2 Hardware description

In this project the following components have been used:

- Raspberry Pi Pico H
- A 42HB34F08AB Stepper Motor
- A A4988 Stepper Motor Driver
- A 1602 LCD with I2C interface
- A 4x4 button matrix

The Raspberry Pi Pico has the following key features:

- The Raspberry Pi Pico uses the RP2040 microcontroller chip, which features a dual-core Arm Cortex-M0+ processor.
- Memory
    - 264kB of on-chip SRAM in six independent banks (four 64 KB, two 4 KB)
    - supports up to 16MB of external flash memory.
- I/O
    - 30 GPIO pins, 4 can be used as analog inputs.
- Peripherals
    - 2 UARTs
    - 2 SPI controllers
    - 2 I2C controllers
    - 16 PWM channels
    - 1 USB 1.1
    - 1 PHY
- The system does not possess internal flash or EEPROM memory. Following a reset, the bootloader is designed to load firmware from either an external flash memory or USB into the internal SRAM. To reset the system, you need to press the BOOT button while connecting the device via USB.

The A4988 Stepper Motor Driver facilitates the management of the motor's phases. This is achieved by setting the intended direction through the DIR pin and generating pulses on the STEP pin equal to the number of steps you want to move the motor. Although not used in this project, the driver also allows for micro stepping by configuring the MS1-3 pins.

The 42HB34F08AB Stepper Motor is a standard stepper motor with 200 steps.

The 1602 LCD with I2C interface, is a standard LCD with an I2C interface, after initialization by the Pico, the cursor position and the displayed text can be configured by sending the propriate messages to the LCD.

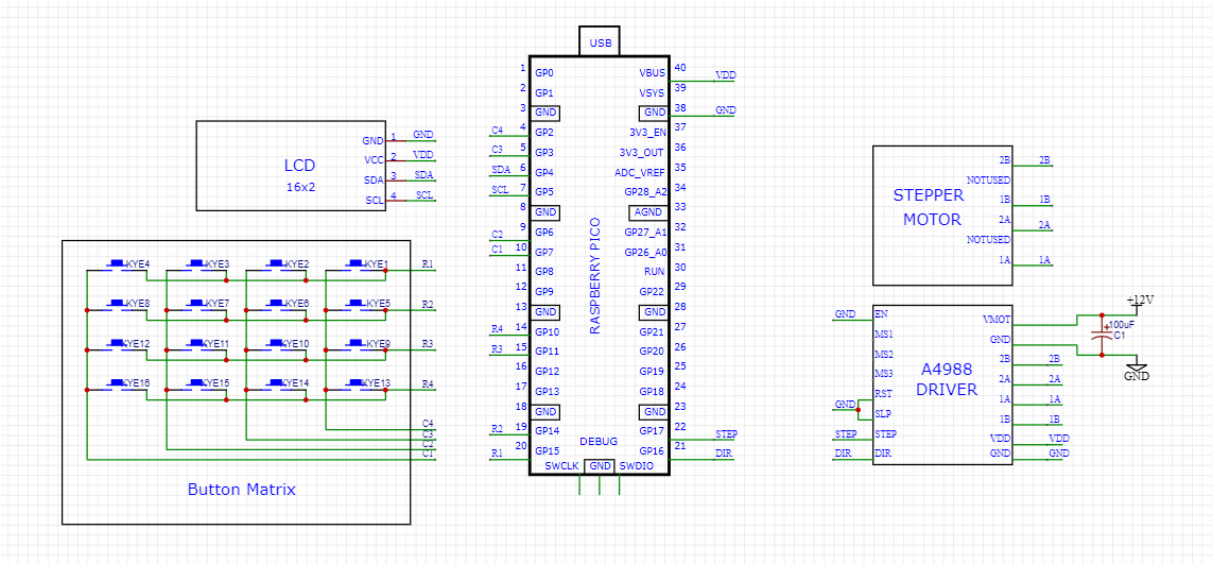

Figure 1: Schematic diagram of the project

In figure 1, the schematic diagram of the project can be observed. It defines how the different components of the project are interconnected.

The LCD:

- The GND pin is connected to the GND of the Raspberry Pico.
- The VDD pin is connected to the VBUS of the Raspberry Pico.
- The SDA pin is connected to the GP4 pin (physically, pin 6) of the Raspberry Pico.
- The SCL pin is connected to the GP3 pin (physically, pin 7) of the Raspberry Pico.

The Button Matrix:

- R1 is connected to the GP15 pin (physically, pin 20) of the Raspberry Pico.
- R2 is connected to the GP14 pin (physically, pin 19) of the Raspberry Pico.
- R3 is connected to the GP11 pin (physically, pin 15) of the Raspberry Pico.
- R4 is connected to the GP10 pin (physically, pin 14) of the Raspberry Pico.
- C1 is connected to the GP7 pin (physically, pin 10) of the Raspberry Pico.
- C2 is connected to the GP6 pin (physically, pin 9) of the Raspberry Pico.
- C3 is connected to the GP3 pin (physically, pin 5) of the Raspberry Pico.
- C4 is connected to the GP2 pin (physically, pin 4) of the Raspberry Pico.

The A4988 Driver:

- The EN, RST, SLP pins are connected to GND.
- The STEP pin is connected to GP17 (physically, pin 22) of the Raspberry Pico.
- The DIR pin is connected to GP16 (physically, pin 21) of the Raspberry Pico.
- The VMOT pin is connected to a 12V source.
- The 2B pin is connected to the 2B pin of the stepper motor.
- The 1B pin is connected to the 1B pin of the stepper motor.
- The 2A pin is connected to the 2A pin of the stepper motor.
- The 1A pin is connected to the 1A pin of the stepper motor.
- The VDD pin is connected to the VBUS of the Raspberry Pico.

Of note are the following design considerations:

1. The pins connected to the button matrix from the Raspberry Pico were chosen such that there was no overlap between the default I2C pins. The pins were also chosen to be far apart from each other to have an easier time when diagnosing issues.
2. For the A4988 Driver:
    a. A heatsink is used on the driver's chip because the rapid commuting of the phases required to control the motor dissipates a large amount of heat.
    b. The driver requires an additional source of power for the stepper motor. As the motor requires a voltage of 12V and an intensity of 1A
    c. An electrolytic capacitor is between the 12V and GND of the additional power source to alleviate power spikes.

3 Software description

In order to develop C code for the Raspberry Pi Pico, the following setup documentation has been used.

The software is separated into 4 distinct parts, each responsible for a separate part of the project. Namely these are:

1. The software that is responsible for managing the button matrix
2. The software that is responsible for managing the LCD display
3. The software that is responsible for managing the Stepper Driver
4. The software that is responsible for the overall control of the parts mentioned above

3.1 Button Matrix Software

The Button Matrix Software does the following:

1. It designates the R1-4 pins as outputs and are set to high.
2. It designates the C1-4 pins as inputs and sets interrupts for RISING EDGE.
3. It detects the column of a pressed button via a RISING EDGE interrupt on pins C1-4.
4. Once a column is known, the row is determined by sequentially setting the row pins to LOW, checking if the column input has changed from LOW to HIGH.
    a. If the input is still HIGH, then the current row pin is restored to HIGH and the next is set to LOW.
    b. If the input is LOW, then we have found the row of the pressed button.
    c. With the values for the column and the row we can determine the value of the pressed key
    d. Because of the use of interrupts, we use a global queue to store the key values so they can layer be used by the main program

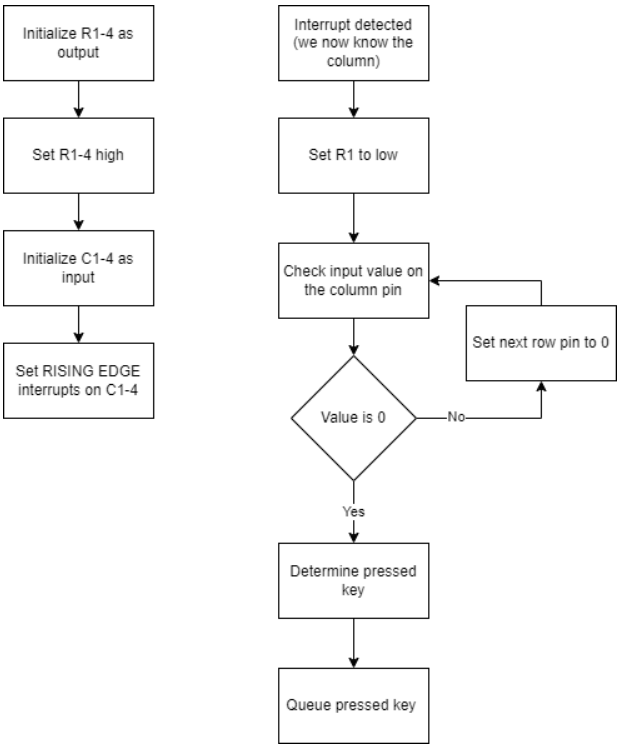

Figure 2: Button Matrix Software Flowchart

3.2 LCD display software

For the LCD display software, the example used for the I2C LCD found in the Raspberry Pi Pico SDK Examples is heavily relied upon.

The code provides examples of how to initialize the correct pins, set the baud rate and provides code that can set the cursor to a desired location and functions that can send strings to the LCD.

3.3 Stepper Driver Software

The control of the stepper driver is done by setting the DIR pin to either 0 or 1, depending on the desired rotation and by pulsing the STEP pin the desired number of steps.

3.4 Overall Control Software

The job of control software is to bring together the parts described in the chapters above.

It achieves this goal in the following manner:

- The necessary systems are initialized.
- Characters from the button matrix queue are dequeued.
- The current command is shown on the LCD.
- Once the EXECUTE KEY is dequeued it starts to parse the text found before the EXECUTE KEY.

- The text is parsed into 3 possible commands and their associated number. Any sequence of commands is possible. The commands are:
    - R[number] – for movement to the right by [number] steps
    - L[number] – for movement to the left by [number] steps
    - B[number] – for changing the delay between two movement commands.
- As the movement commands are executed by the stepper, the command is displayed on the LCD screen.
- The program then awaits new commands.

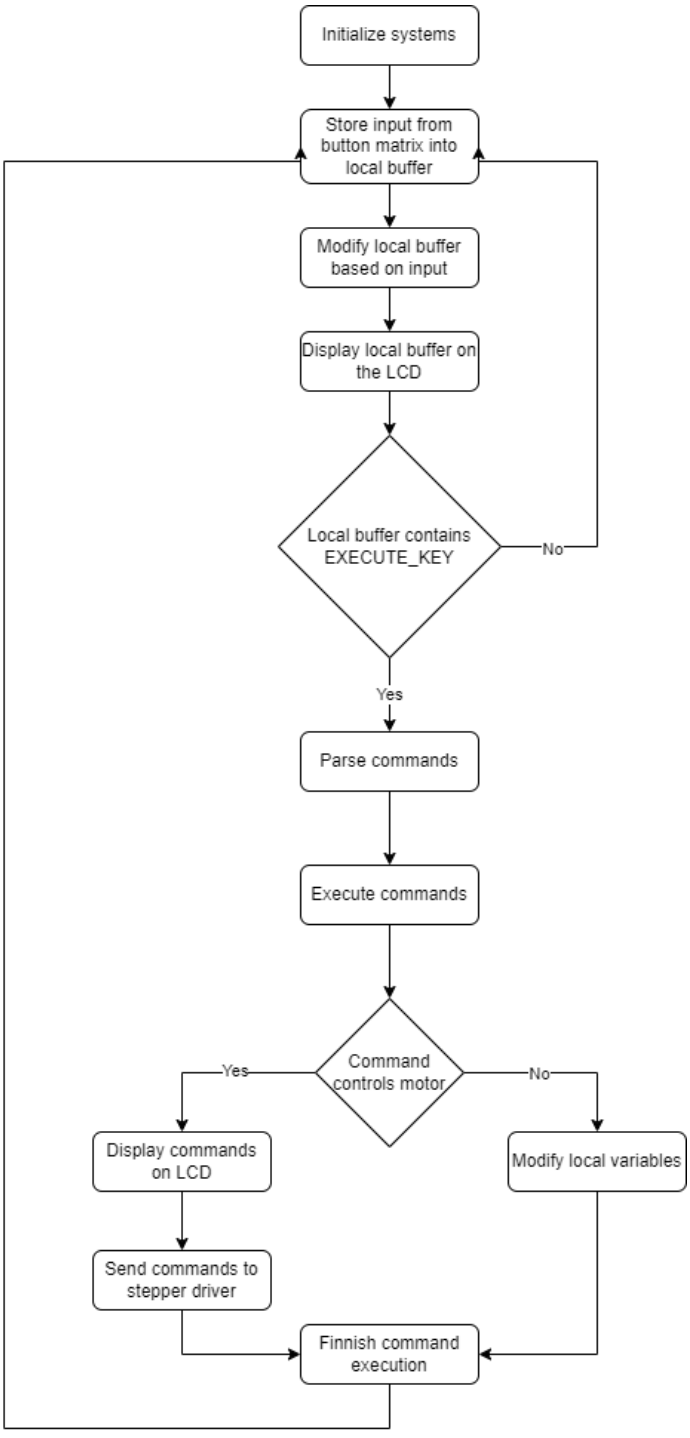

Figure 3 Control Software Flowchart

4 Problems and solutions

The setup mentioned in chapter 3 for the Raspberry Pi Pico is intended to be used from another Raspberry platform to have full access to debugging tools. As this was not applicable in my case, I had to rely on just printing out via serial port and to read it via minicom on the host machine the information I required at that time. Unfortunately, I could not make the example given on pages 14-15 of the setup document to work. My solution was to change the minicom profile to read from /dev/ttyACM0 instead of the default and to just execute the minicom command without other parameters.

5 Conclusions

The project has successfully fulfilled all the stipulated requirements as outlined in the project description. This endeavor has provided me with a valuable opportunity to deepen my comprehension of the Raspberry Pi Pico platform. It has enabled me to harness its capabilities for data acquisition and device control, thereby broadening my technical skill set. Moreover, this project has presented its own set of challenges, each of which I have managed to overcome, thereby reinforcing my problem-solving abilities. These experiences have not only enriched my knowledge but also prepared me to tackle more complex projects in the future.