**Operation of the SAMH**

This section discusses the operational aspects of the system. To minimize the number of programs and systems that the user must interact with, the system is operated entirely using MATLAB, which is needed anyway to perform the measurements with the NI CDAQ. To this end, MATLAB is used to establish a direction SSH connection between the operating PC and the Raspberry Pi, such that MATLAB can trigger the operation of the motor through the SSH connection. At the start of the MATLAB script, the user inputs the sampling rate; the number, type, and sensitivities of the measurement channels; and the desired accuracy for the applied force. Next, the code asks the user if calibration of the system is needed. The system is considered calibrated when the zero position is set such that the hammer is contacting the structure without substantially deforming it. The final step before measurements can be performed is to designate which hammer tip is being used (black rubber tip or white plastic tip), the number of experiments that the user wants to perform, and duration of the measurement. With these inputs set, the user specifies the desired force amplitude, then the system commences the task of exciting the structure. The system iterates on the inputs to the motor until the desired force level is achieved, then the measured force and acceleration responses are saved to a file specified by the user. Note that in this work, we set the maximum number of iterations to 20, but only six iterations were needed at most for the system to produce a force within the desired range, except for in the convergence study discussed in Section 3.4. However, the user can define the maximum number of iterations to fit their needs. A flowchart of the components of the system and how they connect together is presented in Figure 1. The full MATLAB and Raspberry Pi codes are available in the same GitHub directory.



Figure 1. Flowchart representation of the proposed system.

## **System Configuration and initial setup**

In this subsection, we delve into the setup and integration of the system components, focusing on several key aspects to ensure operation and communication within the system. First, we outline the process of connecting the stepper motor and the serial communication module to the Raspberry Pi, a crucial step for initiating precise control and data exchange. Next, we explore establishing an SSH (Secure Shell) connection between a PC and the Raspberry Pi, allowing for remote access and control. Lastly, we discuss the connection and configuration of NI CDAQ system. This setup enables the integration of various system components, ensuring they operate in harmony and communicate effectively.

### **Stepper Motor, Serial Communication Modul Connections to Raspberry Pi**

Figure 2 presents the GPIO (General Purpose Input Output) map for the Raspberry Pi 4 Model B. Any of the GPIO pins can be designated in software as an input or output pin and used for a wide range of purposes. Within our system architecture, a mere pair of GPIO pins suffices to establish a connection between the stepper motor driver and the Raspberry Pi. Specifically, we have selected GPIO 12 and GPIO 18, which are hardware PWM capable, for this purpose. These pins are represented in our Python script as **driverPUL** and **driverDIR**, respectively. Importantly, users are granted the flexibility to modify these pin assignments in the code should they opt for alternative GPIO connections.



Figure 2. Raspberry Pi 4 Model B, GPIO Map

To establish serial communication between MATLAB and a Raspberry Pi, the first step is to connect the communication module to the Raspberry Pi. This module comes equipped with Tx (Transmitter) and Rx (Receiver) pins, essential for data transmission. For successful data exchange, it is necessary to connect the module's Tx pin to the Raspberry Pi's Rx pin, and the module's Rx pin to the Raspberry Pi's Tx pin. It is crucial to note that Raspberry Pi GPIO pins are designed to accommodate a maximum voltage of 3.3 V. Therefore, to prevent damage, ensure that any connected module is set to operate at 3.3 V, typically achieved by adjusting a voltage-level jumper on the module to 3.3 V.

The Raspberry Pi 4 supports several UART (Universal Asynchronous Receiver-Transmitter) interfaces, enabling versatile serial communication capabilities. UART is a highly flexible protocol suitable for a range of applications, including interfacing Raspberry Pies with each other, connecting to Arduino boards, or establishing communication with PCs. In our project, we are utilizing one of the available UART interfaces, which necessitates careful selection and configuration of GPIO pins for communication.

While the default configuration for UART communication on the Raspberry Pi uses GPIO 14 (TXD) and GPIO 15 (RXD) for UART0, it is possible to configure additional UARTs on other GPIO pins as required. It is important to remember that by default, the primary UART (UART0) is often allocated to the Bluetooth module on Raspberry Pi models 3 and 4, and a mini UART (UART1) is made available for general purposes. However, this setup can be modified according to the user's needs.

To activate and use additional UART interfaces beyond the default setup, one must append the line **dtoverlay=uartx**—with x representing the UART number—to the **/boot/config.txt** file on the Raspberry Pi. This step is crucial for enabling the desired UART interface for your application. In our case, we chose to use UART2 which corresponds to GPIO 0, and GPIO 1. One may check the documentation for all the alternatives.

Finally, ensuring that the correct serial port is selected for communication between MATLAB and the Raspberry Pi is essential for effective data exchange. In our configuration, **COM3** is identified as the appropriate port on Windows systems, while **/dev/ttyAMA1** is on the Raspberry Pi. This correct port identification is vital for establishing a successful serial connection, allowing MATLAB and the Raspberry Pi to communicate seamlessly.

### **IP configuration for SSH and NI CDAQ**

Establishing an SSH connection from MATLAB to a Raspberry Pi is straightforward, requiring only the Raspberry Pi's IP address, username, and password. If the Raspberry Pi is connected to Wi-Fi, acquiring its IP address for SSH access is relatively simple. However, security considerations and the potential for the IP address to change when connecting to different networks pose challenges, as it necessitates retrieving the IP address anew with each connection attempt, which is far from ideal.

To circumvent these issues, we opt for a more stable solution by establishing an SSH connection through a LAN (Local Area Network), using a static IP address for the Raspberry Pi. This approach requires the PC running MATLAB to have two LAN ports: one for SSH access to the Raspberry Pi and the other for communication with the NI CDAQ. When using a LAN connection, it is imperative to manually assign IP addresses to ensure that all devices are configured correctly and can communicate effectively. An essential aspect of this setup is that all IP addresses must belong to the same subnet to ensure seamless communication between devices. Private IP address ranges reserved for local networks include 192.168.0.0 to 192.168.255.255, 172.16.0.0 to 172.31.255.255, and 10.0.0.0 to 10.255.255.255. Selecting an IP address within these ranges for the Raspberry Pi and ensuring that the IP addresses of other devices (such as the PC and NI CDAQ) are within the same range, is crucial for establishing a reliable network configuration. In our configuration, for example, we assigned the IP addresses as follows: 192.168.1.10 for the Raspberry Pi, 192.168.1.11 for the PC, and 192.168.1.12 for the NI CDAQ, thereby guaranteeing smooth inter-device communication.

Moreover, for enhanced security and convenience, it is recommended to generate an SSH key on the PC running MATLAB and copy this key to the Raspberry Pi. This step eliminates the need to enter the Raspberry Pi's password every time an SSH connection is initiated from MATLAB, streamlining the process and bolstering security.