UNIVERSITY OF CALIFORNIA
SANTA CRUZ

VALVE CONTROL SYSTEM BY MULTIPLE DIGITAL SIGNAL PROCESSING

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER of SCIENCE

in

ELECTRICAL AND COMPUTER ENGINEERING

by

Ruiming Zhao

June 2024

Table of Contents

List of Figures

List of Tables

Abstract

Valve control system by multiple digital signal processing

by

Ruiming Zhao

Nowadays, signal transmission and control have become crucial. The discussion revolves around how to control more efficiently, rapidly, and economically. Specifically, for instance, the digital signals needed to control a switch valve, where 1 represents open and 0 represents closed. The overall experiment requires a significant number of digital signals. This article will delve into how to construct the switch controls for numerous valves and integrate it into automated control applications.

In biological research and experiment, valve control systems are frequently used. For example, a low-cost microscope named 'Picroscope', which can be used for simultaneous and longitudinal biological imaging, is reported. Programmable valve controls system is pivotal. In this work, efficient electronic circuits and software controls are studied and built for this similar purpose with more cell plates and less bulky hardware components at low cost.

Our valve control system consists of a Raspberry Pi microcomputer, the Adafruit PCA9685 16-channel servo driver boards, and the servo motors. Using these hardware components that process through I^{2}C protocol, the valves can be adjusted in a general way by python programming on the microcomputer. Namely, the valves can be opened with any angle for arbitrary duration of time and closed at any time. With two I^{2}C protocol pinouts and multiple secondaries (Adafruit) circuit board, the entire system will be able to handle thousands of valves.

Keywords: Valve control, Raspberry Pi, servo motor, I^2C protocol

Thanks to

Myself,

My great teacher Mircea,

professor Hao Ye and professor Aviv Elor,

My parents and the opportunity studying at UCSC.

Chapter 1

Introduction

Laboratory robotics encompasses the utilization of robots and automation technologies within lab settings to either conduct or aid in tasks traditionally performed by human researchers and technicians. Such technologies are increasingly applied across diverse fields including biology, chemistry, pharmaceuticals, and environmental science, enhancing efficiency, precision, and safety. Automated liquid handling systems, robotic arms for sample manipulation, automated incubators, and comprehensive systems capable of executing a series of tasks autonomously are notable examples of laboratory robotics.

The focus of this thesis is the design of an automated valve control system aimed at managing a large number of valves for the input and output of media. This system encompasses both circuitry and practical deployment, emphasizing the installation of a suitable system for biological experimentation. It involves the use of appropriate digital and Pulse Width Modulation (PWM) signal processing, alongside the development of software controls, requiring expertise in both software and hardware domains.

1.1 Motivations

The prolonged observation and recording of tissue cells and other living specimens are pivotal in biological research. Conducting extended duration experiments necessitates the injection and removal of biological culture fluids into and from sample plates. For experiments focusing on cell development, it is essential to utilize circuit boards that are adequately sized, well-connected, and functionally optimal. Oversized circuit boards can consume excessive experimental space, especially in studies requiring hundreds or even thousands of digitally controlled output ports. Moreover, logically organized and clear wiring on the circuit board is critical for facilitating swift modifications or wire management in subsequent stages. The functionality of a circuit board, defined by the absence of superfluous electronic components, plays a significant role. Thus, optimizing the overall circuit design to minimize costs is also a key objective.

1.2 Objectives

The objectives of this thesis work are:

• Designing a fully programmable valve control system for long-term biological research applications.

• Prototyping hardware components that are easy to build, small sized and compact, and have low cost and good functionality.

- Coding with python programming, I^{2}C protocol, and hardware-aimed software packages for the valve system controls.

- Testing experiment and demonstrating the effectiveness of the valve control system.

Chapter 2

Literature Review

Capturing and analyzing the dynamics and growth of live tissue cells are central to experimental biology and biomedical research [1]. Conducting imaging within incubators presents numerous benefits over external imaging for live cell cultures, attributed primarily to the incubator's ability to sustain a controlled environment crucial for the live cells' well-being and accurate behavioral study during experiments. Key benefits of in-incubator imaging include a stable environment for cells, the capability for continuous observation, a lower risk of contamination, and enhanced viability of cells. The progress in laboratory robotics and automation is revolutionizing the way cell culture experiments are conducted, facilitating more intricate, accurate, and high-throughput studies while minimizing manual intervention and the likelihood of errors.

Recent innovations include the 'Picroscope', a 3D printed, cost-effective microscope designed for both simultaneous and longitudinal biological imaging [2]; an open-source, pneumatically controlled setup for operating single and multi-layer microfluidic devices with automated scripting [45]; and a fully integrated system for managing pneumatically-actuated microfluidic chips in biological research [41]. These technologies offer significant potential for diverse applications in imaging and recording under various conditions, with the possibility of integrating new functionalities. Specifically, there's a notable interest in tracking the growth and behavior of whole organisms or individual cells over extended periods, necessitating long-term imaging and recording. This requires not only the culture fluid injection for observing biological specimens but also the development of automated valve control systems alongside the automation of sliding stages and imaging cameras.

In advancing biological research methodologies, this work aims to explore and construct an efficient valve control system comprising effective electric circuit boards and software controls. This system is intended to accommodate a greater number of cell plates while reducing the physical size of hardware components compared to existing systems. This review first outlines previous developments in this field before detailing our contributions, which are categorized into three main areas: (1) Imaging and recording systems tailored for biological research; (2) Integration and optimization of hardware components and software for control processes; (3) A comprehensive overview of the achievements to date.

2.1 Imaging and capturing pictures from microfluidics environment

The progression in microscopic imaging has been a catalyst for breakthroughs in biological research, enabling detailed observations of cellular dynamics, organismal development, and behavioral patterns [1]. Essential attributes of an effective imaging system include rapid image capture, non-intrusive observation that leaves the specimen unharmed, adaptability to diverse environmental conditions such as temperature, lighting, and humidity, high resolution, compact design, and affordability [2].

The focus on cost-effective commercial live cell imaging systems has grown, aiming to make them accessible to academic and instructional labs worldwide [5, 6]. Innovations leveraging 3D printing and open-source software have markedly expanded the availability of low-cost biomedical instruments for researchers [7, 8, 9, 10]. Nonetheless, the design of most economical 3D-printed microscopes does not cater to the demands of long-term and multi-well culture imaging required for extensive biological studies [11, 12, 13].

The capability to image biological specimens both simultaneously and over time holds significant value for biological research. Such imaging is pivotal for endeavors like drug discovery, genetic analysis, and extensive phenotyping to understand biological mechanisms and diseases [14, 15, 16]. Two budget-friendly approaches have emerged: employing a single movable camera [17, 6, 5] and utilizing a wide-field camera with subsequent post-processing [18, 19]. The 'Picroscope' system stands out by providing

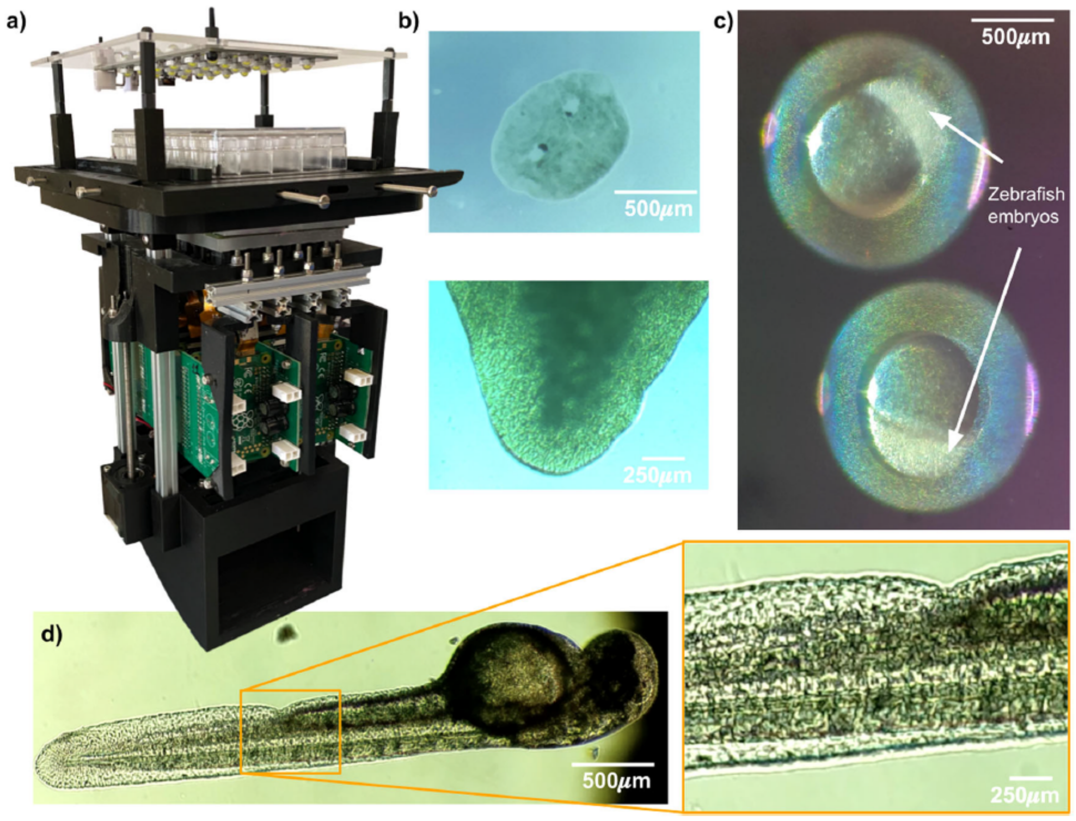


Figure 2.1: Development of Picroscope, a low-cost system for simultaneous longitudinal biological imaging. (a) The Picroscope fits a standard 24-well plate, it is controlled remotely and images can be accessed through a web browser. (b–d) Applications of the Picroscope to longitudinal imaging of developmental biology and regeneration. [2].

a practical solution for automated imaging across a 24-well cell culture plate using 24 individual lenses, showcasing the device's versatility in capturing various biological samples [2]. Figure 2.1 illustrates the Picroscope setup and the array of biological subjects it can image.

Conversely, commercial microscopy systems developed for the concurrent imaging of biological specimens often focus on cells arranged in monolayers [20]. There has been a notable focus on the longitudinal observation of entire organisms in three-

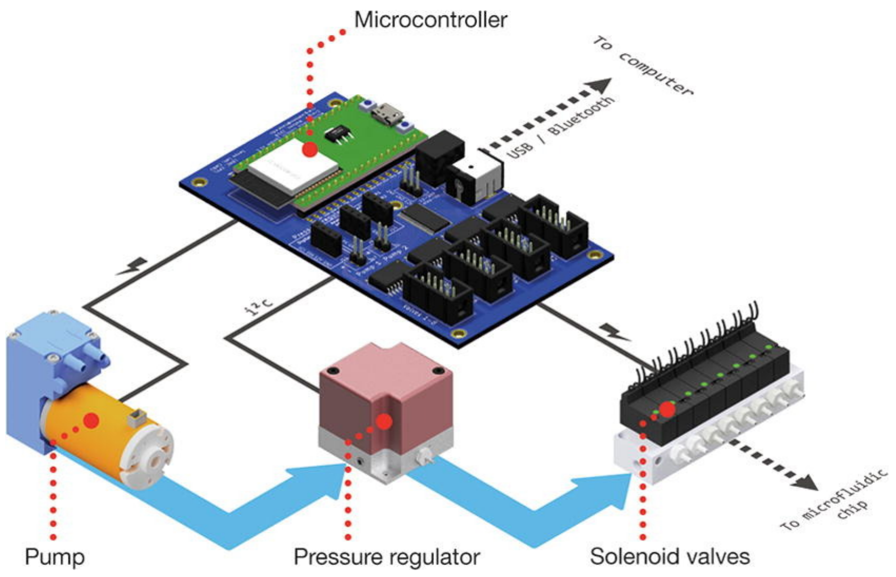


Figure 2.2: The pressure regulators were purpose-built, enabling integrated pressure sources on-board, rather than relying on external equipment. These regulators can also be used as stand-alone devices in any other application [41].

dimensional contexts [21, 22, 23]. The Picroscope stands out by offering the capability to capture images not only on the xy plane but also along the z-axis. This functionality is achieved through precise control provided by two stepper motors, which adjust the height of an elevator unit that supports all 24 camera objectives [2].

The monitoring and documentation of neuronal activity are crucial for observing neural dynamics, playing a significant role in various biological and biomedical research areas. Extracellular recording techniques have gained prominence as an effective method for assessing neural activity without harming or altering the tissue cells under observation [24, 25]. Persistent and longitudinal recordings are vital for analyzing responses to electrical or pharmacological stimuli and for tracking neurodevelopmental patterns and dynamics over time [26, 27, 28].

An integrated automated microfluidics control system, leveraging pneumatically-operated microfluidic chips for solution management, has been developed [45]. Such chips often necessitate connecting numerous tubes to deliver reagents and to activate on-chip elastomeric valves. The primary role of these pneumatic microfluidics controllers is to independently modulate multiple valves, allowing for precise pressure adjustments across various tubing lines connected to a microfluidic device. Refer to figure 2.2. The deployment of computer-controlled solenoid valves offers a method to automate the majority of experimental procedures, with numerous implementations of such systems being introduced in recent literature [45, 42, 43, 44].

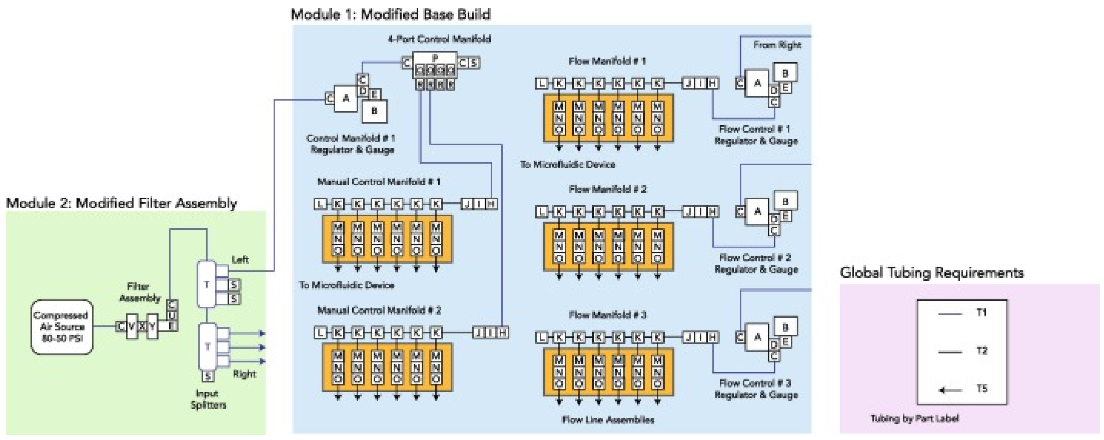


Figure 2.3: A fully manual build for 18 flow lines and 12 manually addressable control lines without automation capabilities which can be used to save costs. [45].

In addition to the system depicted in figure 2.2, two alternative configurations are proposed, as illustrated in figures 2.3 and 2.4. The first alternative presents a purely manual setup with 18 flow channels and 12 control lines that can be manually operated, lacking automation to reduce costs. The second configuration offers a simplified, fully

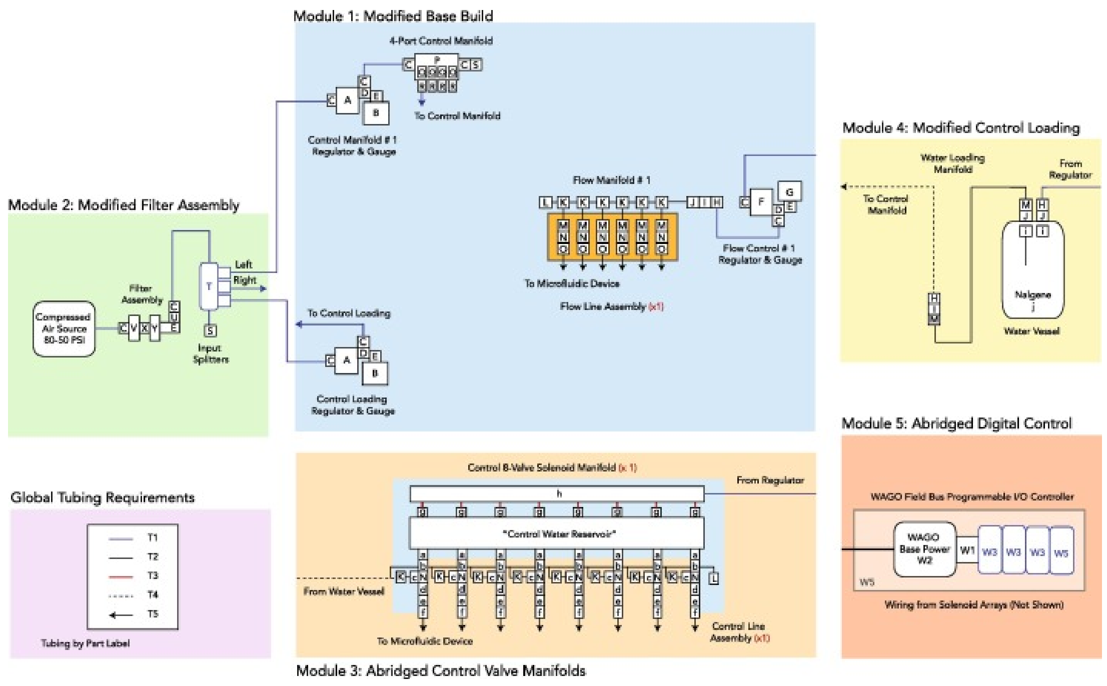

Module 1: Modified Base Build
4-Port Control Manifold
To Control Manifold
Control Manifold # 1 Regulator & Gauge
Flow Manifold # 1
To Microfluidic Device
Flow Control # 1 Regulator & Gauge
Flow Line Assembly (x1)
To Control Loading
Control Loading Regulator & Gauge
Module 2: Modified Filter Assembly
Compressed Air Source 80-50 PSI
Filter Assembly
Left
Right
Input Splitters
Module 4: Modified Control Loading
Water Loading Manifold
From Regulator
To Control Manifold
Nalgene
Water Vessel
Module 5: Abridged Digital Control
WAGO Field Bus Programmable I/O Controller
WAGO Base Power W2
Wiring from Solenoid Arrays (Not Shown)
Global Tubing Requirements
T1
T2
T3
T4
T5
Tubing by Part Label
Control 8-Valve Solenoid Manifold (x 1)
From Regulator
"Control Water Reservoir"
From Water Vessel
To Microfluidic Device
Control Line Assembly (x1)
Module 3: Abridged Control Valve Manifolds

Figure 2.4: An abridged build for 6 flow lines and 8 individually addressable solenoid valves to control 8 control lines that fully automated, programmed and remotely operated for simpler devices. [45].

automated system with 6 flow channels and 8 solenoid valves, each controlling an individual line, designed for straightforward devices that can be programmed and operated remotely [45].

These systems, along with their variations, are adaptable to various environmental settings, with the flexibility to integrate new functionalities as needed. A significant area of focus is the prolonged imaging and analysis of biological samples, which is pivotal for examining tissue cell growth and their dynamic behaviors. This interest drives the objective to develop a system capable of managing multiple imaging and processing tasks in a straightforward and efficient manner, highlighting its potential impact on future research in tissue and cellular studies.

2.2 Hardware components and software controls

This subsection provides a concise overview of the application, historical development, and producers of the hardware and software components that will be employed in the construction of the valve control system.

2.2.1 Raspberry Pi processor

The Raspberry Pi, a compact board or microcomputer, is developed by the UK-based Raspberry Pi Foundation [29]. The Raspberry Pi 4 Model B represents the most recent advancement in the Raspberry Pi series, offering significant improvements in processor speed, multimedia performance, memory, and connectivity over its predecessor, the Raspberry Pi 3 Model B+. It maintains backward compatibility and operates at a similar power level. Key attributes of the Raspberry Pi 4 Model B include its 64-bit quad-core processor, support for dual displays up to 4K resolution through micro-HDMI ports, 4Kp60 hardware video decoding, up to 8GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and the option for PoE (Power over Ethernet) via an additional PoE HAT accessory.

The Raspberry Pi's design avoids the need for extensive physical circuitry, offering plug-and-play circuits that do not necessitate soldering. Components can be easily replaced if issues arise. The device requires a DC power supply, ensuring low power consumption and cost-efficiency for hardware control. Online resources are readily available for assistance and application downloads [31, 32].

In terms of control, the system relies on Python programming, utilizing both custom-written functions and existing library features for software management. Python, an interpreted, high-level, general-purpose programming language since 1991, ranks among the most popular and rapidly expanding programming languages today. The ”Pi” in Raspberry Pi denotes ”Python Interpreter,” emphasizing the language's preferred status on this platform. Python's interpreted nature allows for the interactive execution of commands without the need to compile a program beforehand. Programs are compiled at runtime into an intermediate bytecode, which is then executed by a virtual machine [40].

2.2.2 Adafruit PCA9685 16-channel servo driver

The Adafruit-manufactured board is specifically crafted for managing servo motors via continuous PWM signals [33]. It is frequently employed in the creation of robots, kinetic art with numerous moving components, or arrays of LEDs requiring precise PWM control. The board's dimensions are 62.5mm x 25.4mm x 3mm, and it operates 16 PWM outputs using just two control pins. Additionally, it supports daisy-chaining up to 62 such boards, enabling control of up to 992 PWM outputs. This functionality is facilitated by an I^2C-controlled PWM driver equipped with an integrated clock mechanism.

Included with the product are a fully tested and assembled breakout board, four units of 3x4 male straight headers for connecting servos or LEDs, a 2-pin terminal block for power connection, and a 6-pin 0.1" header for breadboard integration. While assembly requires some light soldering to attach the preferred headers, this process is straightforward and can be completed in approximately 15 minutes, even by novices.

2.2.3 NXP OM13541 (PCAL6534) circuit board

The OM13541, produced by NXP Semiconductors N.V., is a 34-bit GPIO Daughter Card designed for seamless integration with the OM13260 Fm+ I^2C bus development board [34]. This auxiliary board simplifies the process of testing and designing systems that incorporate the PCAL6534. The PCAL6534 is an ultralow voltage, 34-bit, general-purpose I/O expander with voltage-level translation capabilities, enabling remote I/O expansion for a broad spectrum of microcontroller families through the Fast-mode Plus (Fm+) I^2C-bus interface. The PCAL6534 34-bit GPIO evaluation board facilitates bidirectional voltage-level translation and GPIO expansion across a voltage range of 0.8 V to 3.6 V for SCL/SDA and 1.8 V, 2.5 V, 3.3 V, 5.5 V for I/O ports. It features an active low reset input, an open-drain active low interrupt output indicator (red LED), and a hardware address input that enables the selection among four different secondary addresses. Additionally, a graphical interface is provided to allow users to conveniently access and manipulate the functionalities of the I/O expander, which communicates with the host system through a standard I^{2}C-bus/SMBus interface.

2.2.4 Servo motor

A servomechanism, or simply servo, is a term that describes a variety of machine types that have been in existence for longer than many might expect [35]. At its core, a servo is a motorized system that includes a feedback component. This system typically comprises a DC motor, a control circuit, and a potentiometer or another type of feedback device integrated into a conventional servo setup. The DC motor is connected to a gearbox and an output or drive shaft, enhancing the motor's speed and torque. The control circuit processes signals from the controller, while the potentiometer provides feedback to the control circuit, allowing it to track the position of the output shaft. Hobby servos almost universally feature a standard three-pin, 0.1" spacing connector for both power and control purposes. Although color coding for these pins may differ across brands, the configuration of the pins is largely standardized. This arrangement enables the control of the direction, speed, and position of the output shaft using merely three wires. The specific servo motor discussed in this research is produced by SparkFun Electronics [35].

Chapter 3

Problem setup for the valve control system

In the context of extended-duration experiments involving the imaging and recording of tissue cells or other biological specimens, there is a critical need to culture these samples under specific conditions over designated periods. As depicted in Fig. 3.1, this necessitates the precise injection of culture fluid into the sample plate in predetermined quantities and for set durations, followed by its timely removal. Achieving this cyclical process requires the development of a programmable mechanism for the precise opening and closing of valves, facilitating both the injection and extraction of fluids.

In this chapter, we introduce and develop an automated, programmable valve control system designed to handle an increased volume of sample cells while minimizing the physical footprint of hardware components. A schematic representation of the sys-

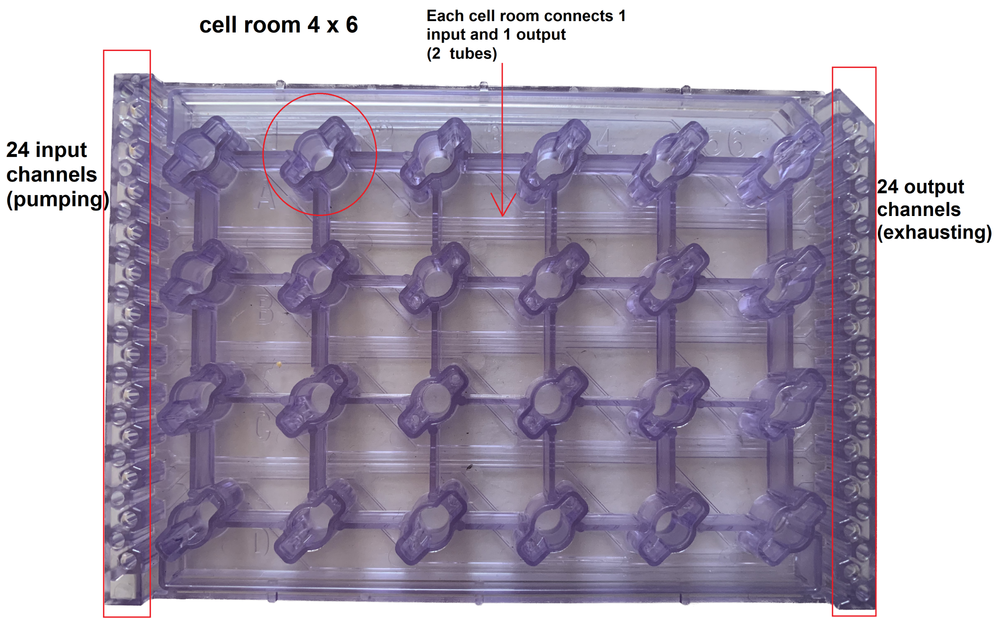

cell room 4 x 6
Each cell room connects 1 input and 1 output (2 tubes)
24 input channels (pumping)
24 output channels (exhausting)

Figure 3.1: 24 culture plates sample holder for biological imaging and recording experiments. The 24 plates locate in the central large area. There are 24 holes on both the left and right sides for the injection and suction of the culture fluid, respectively. Each holes are connected to a valve controlled by a servo motor with programmable opening or closing operations. The culture fluid flows in and out of the plate through micrometer sized tubes.

tem's hardware layout is provided in Fig. 3.4. At the heart of the control system, the Raspberry Pi 4 functions as the primary computing unit, facilitating programming and control tasks. This device can be outfitted with standard peripherals like a monitor, mouse, and keyboard for user interaction. It interfaces with the Adafruit circuit board through an I^{2}C connection, which in turn manages a micro servo motor responsible for actuating the valve. As detailed in subsequent chapters, the precise timing and duration of valve operations are controlled by the servo motor's rotation, enabling the execution of prolonged imaging and recording experiments.

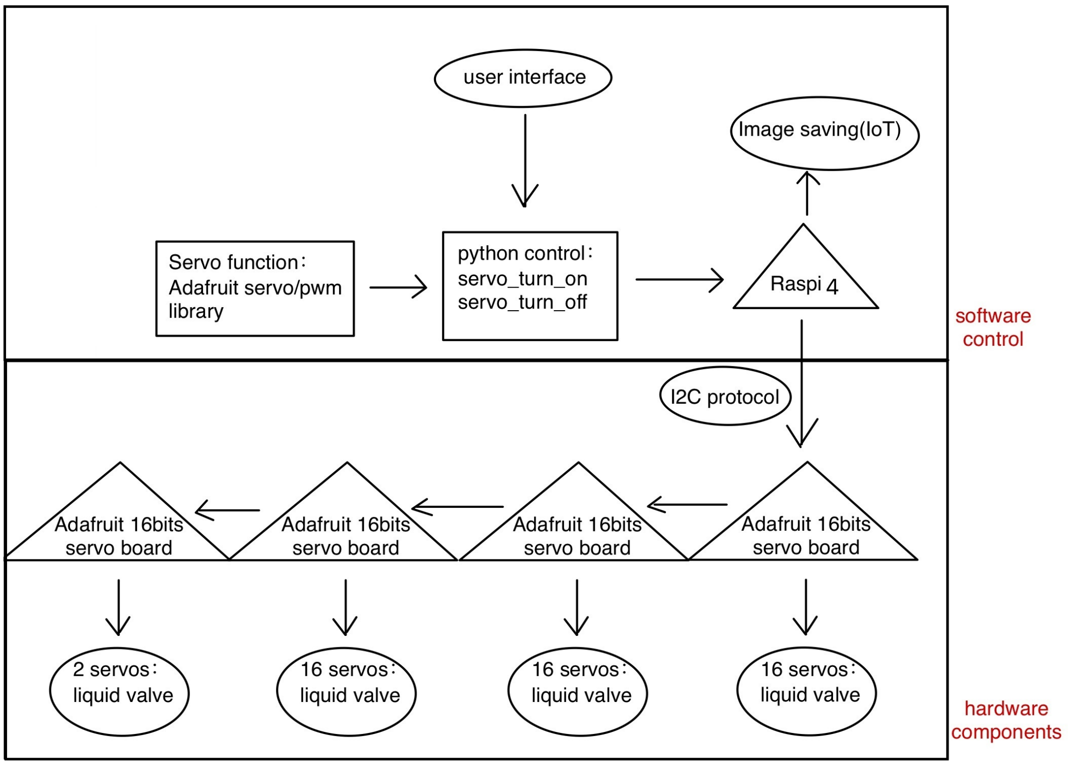

user interface
Image saving(IoT)
Servo function: Adafruit servo/pwm library
python control: servo_turn_on servo_turn_off
Raspi 4
software control
I2C protocol
Adafruit 16bits servo board
Adafruit 16bits servo board
Adafruit 16bits servo board
Adafruit 16bits servo board
2 servos: liquid valve
16 servos: liquid valve
16 servos: liquid valve
16 servos: liquid valve
hardware components

Figure 3.2: Overview of the valve control system (flow chart). As the goal of the present work, here only 50 controlled servo motors are shown.

The system's operational workflow and software control mechanisms are depicted in Fig. 3.2. With the capability to daisy-chain up to 62 Adafruit boards through I^2C secondary cables, as illustrated in Fig. 3.4, and considering the Raspberry Pi 4 processor's dual I^2C connectors, a single Raspberry Pi 4 can, in theory, control up to 1984 motors.

It's important to note that the Adafruit board could be substituted with the NXP OM13541 circuit board, which may significantly diminish the size of the hard-

Table 3.1: Information of the hardware components.

Name	Model	Weight (g)	Size (inch^3)	Cost (USD)
Raspberry Pi	RasPi 4 B	59	3.74×2.76×0.98	35-75
Adafruit	PCA9685 16x12-bit PWM	9	2.5×1.0×0.1	15
NXP	OM13541(PCAL6534)	39.5	6.22×2.09×0.39	173.61
Servo	SG90	9	0.91×0.48×1.14	1.877

ware components required. However, adopting the NXP board might necessitate the development of additional Python code and libraries due to the absence of compatible open-source software packages. The capabilities and features of the NXP OM13541 circuit board will be elaborated upon further in this work, with a concluding discussion on its potential future applications provided in the final chapter.

Table 3.1 details the specifications of the hardware components involved in this system. As outlined in the table, the total cost of assembling the system is projected to remain under one thousand dollars, making it an economically viable option for cell growth experiments. This setup is ideally suited for efficiently managing media changes across each cell on a miniature plate, as shown in Fig. 3.1.

Figure 3.2 illustrates the control of a total of 50 motors within the current study. Additionally, Figure 3.3 depicts the Python-based logic for managing each servo motor, tailored to various functions. Further details on this aspect will be provided in a subsequent chapter.

3.1 What has been done

By synthesizing the hardware and software knowledge previously discussed, we have developed a distinctive control system tailored for managing biological experiments, such as those required for our study. This system not only builds upon the foundations laid by prior research but also facilitates experiments with greater accuracy and streamlined practical execution. The design and implementation of both the hardware and software components are specifically engineered for efficient signal transmission and control, directly catering to the biological research needs outlined earlier. This system is conceived as an extension of the automated culture system, with an emphasis on enhancing valve controls while ensuring the experimental setup remains compact. The forthcoming sections will provide a comprehensive guide on assembling the entire system. This approach aims to bridge the existing divide between electrical engineering and biological research.

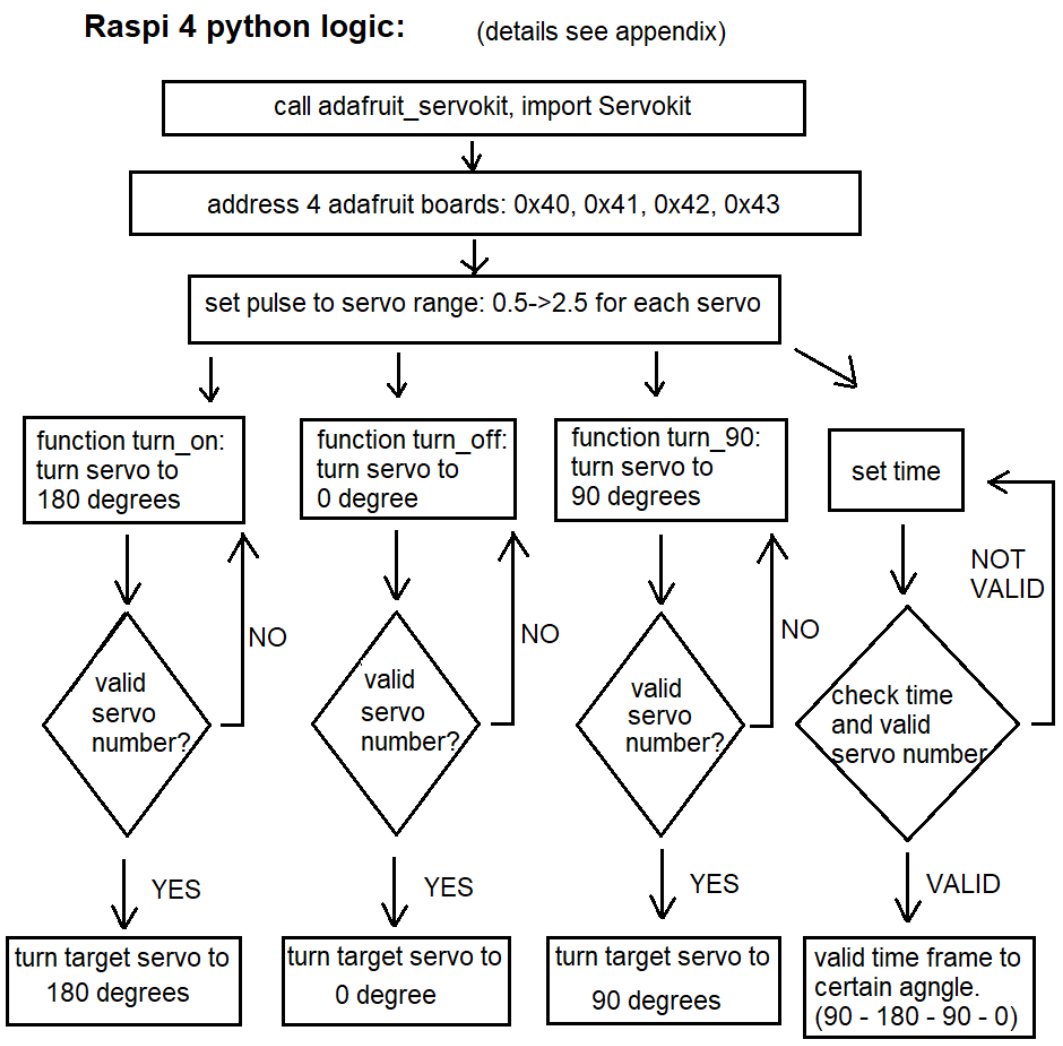

Raspi 4 python logic:
(details see appendix)
call adafruit_servokit, import Servokit
address 4 adafruit boards: 0x40, 0x41, 0x42, 0x43
set pulse to servo range: 0.5->2.5 for each servo
function turn_on: turn servo to 180 degrees
function turn_off: turn servo to 0 degree
function turn_90: turn servo to 90 degrees
set time
valid servo number?
NO
YES
turn target servo to 180 degrees
valid servo number?
NO
YES
turn target servo to 0 degree
valid servo number?
NO
YES
turn target servo to 90 degrees
check time and valid servo number
NOT VALID
VALID
valid time frame to certain agngle. (90 - 180 - 90 - 0)

Figure 3.3: The pseudo-code in python for valves system development.

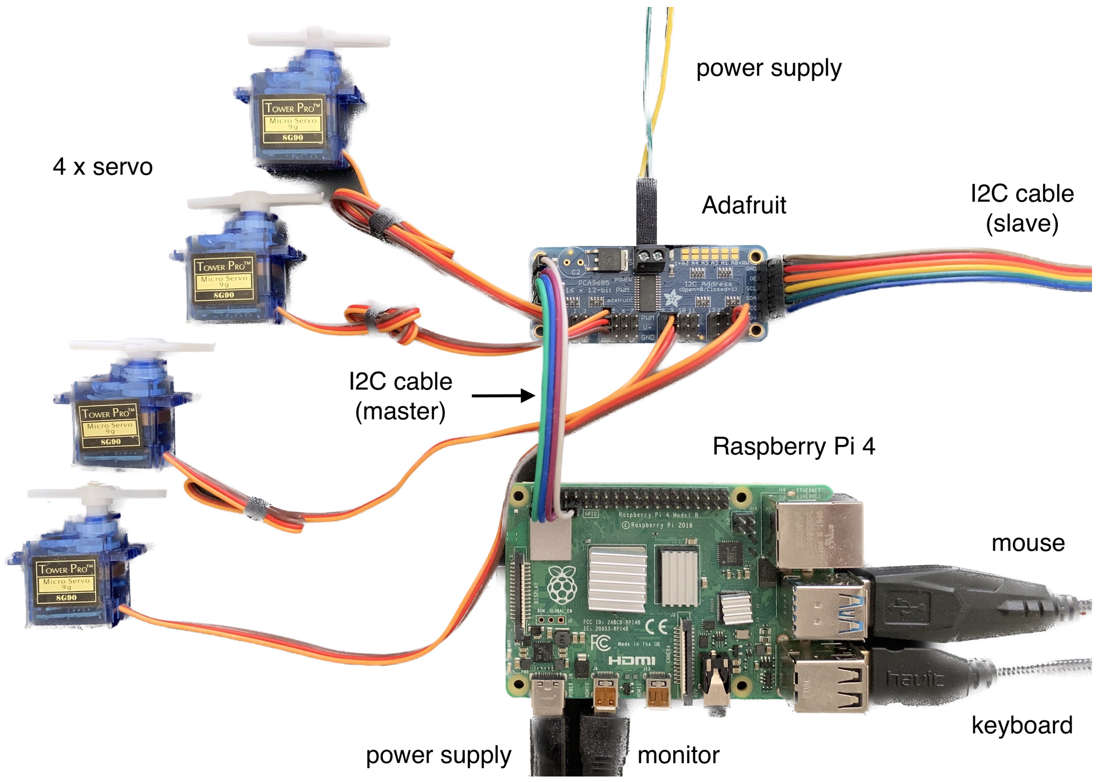


Figure 3.4: Overview of the valve control system (hardware components). A Raspberry Pi 4 microcomputer together with one Adafruit circuit board and four servo motors is shown as an example. The total number of controllable motors may increase to 1984 in principle.

Chapter 4

Hardware development

This chapter will meticulously detail each hardware component depicted in Fig. 3.4. Furthermore, we will delve into the inaugural generation circuit board to trace its historical evolution and examine the NXP OM13541 circuit board in anticipation of future research applications. Additionally, we will highlight and discuss the primary challenges encountered while soldering these circuit boards.

4.1 Raspberry Pi 4 processor

The Raspberry Pi, a microcomputer capable of processing data at speeds comparable to those of a personal computer's CPU, supports standard peripherals like monitors, mice, and keyboards. It has been extensively utilized in numerous biological imaging and recording experiments [2, 4, 5]. For the purposes of this thesis, we employ the Raspberry Pi 4, as illustrated in Fig. 4.1, a model launched by the Raspberry

Foundation in 2019 [29, 31, 32].

This microcomputer is versatile, supporting a range of applications. To leverage its capabilities for this project, we selected Raspberry Pi Linux as the operating system. This enables the installation of necessary applications onto the single-board computer through the addition of specific hardware components and software programming. Crucial to the operation of effective valve controls, a circuit board is required to facilitate communication between the microcomputer and the motor attached to the valve, allowing for programmable control.

4.2 Assembly of circuit boards

During the assembly of circuit boards, several common challenges such as improper solder joint formation, component misalignment, and the use of too much solder can arise. To ensure high-quality soldering, it's crucial to navigate these issues effectively.

To address the problem of solder joints not forming correctly, it's vital to solder at the right temperature, which is generally about 650 degrees Fahrenheit. Securely fix the circuit board in place, use your dominant hand to operate the soldering iron and your other hand to apply the solder. Approach the joint steadily, wrapping the solder fully around it to reduce the risk of incomplete solder joints.

For preventing misalignment, it is advisable to carefully check the circuit

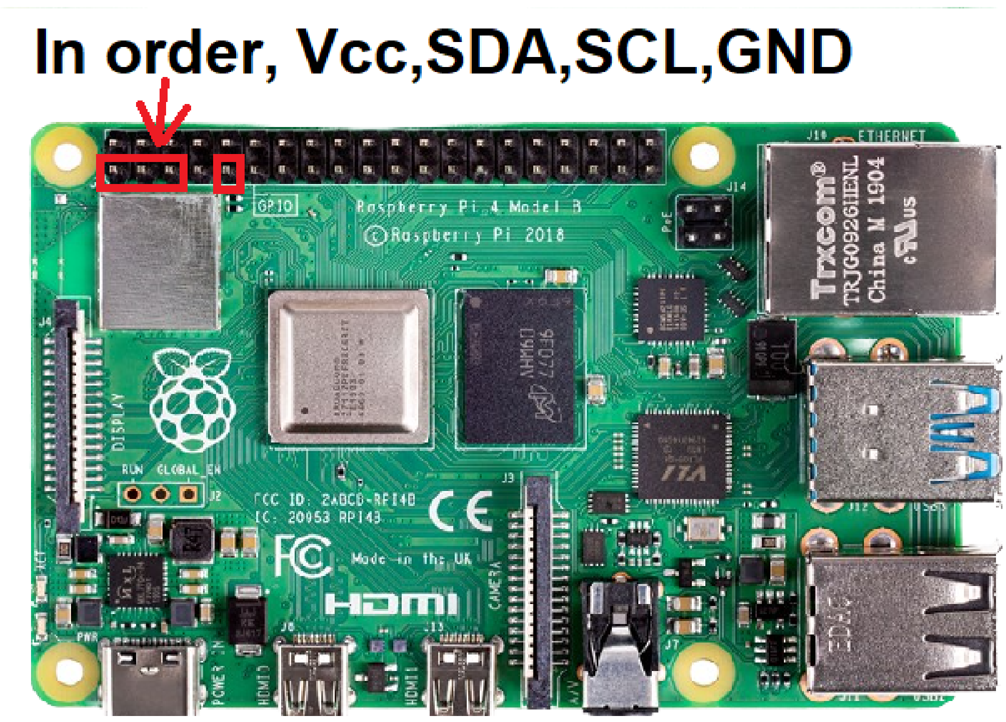


Figure 4.1: Optical photo of Raspberry Pi 4 Model B, a version released by Raspberry Foundation in 2019.

schematics prior to soldering. Pay close attention to ensure that, for instance, adjacent pins on microchips are connected to their correct circuits and not mistakenly soldered together.

To control the issue of excessive solder, be mindful of nearby metal traces when applying solder to a joint. The goal is to use just enough solder to make a secure connection without creating a risk of short circuits by bridging adjacent traces. After the soldering process, employing a multimeter to check for shorts can help ensure the integrity of the circuit.

4.3 The first-generation circuit board

The initial generation of the circuit board was developed to facilitate digital control of switching valves for administering biological culture fluid. In its early iteration, controlling each valve was straightforward, necessitating only a binary signal to toggle between open and closed states. Specifically, a low voltage range from 0 to 0.8V signified a closed valve (signal 0), whereas a high voltage range from 4.6 to 5.4V indicated an open valve (signal 1). Drawing from the design principles established by prior researchers, the circuit board incorporated an MCP23017 (a 16-bit I^2C-compatible I/O expander) and was organized into 13 separate branches. Each branch was equipped with an NPN transistor, a diode, a Zener diode, and a resistor, as illustrated in Fig. 4.2. The construction of the first-generation circuit board, as depicted in Fig. 4.3, evidently involved substantial soldering efforts for affixing transistors, diodes, and resistors.

The microcontroller integrated into this circuit board capitalizes on the Raspberry Pi's I^2C protocol to manage the distribution of digital signals for precise valve control. This method significantly reduces the necessity for multiple pins, making the I^2C protocol an ideal choice for such applications.

Nevertheless, the physical dimensions of the board and the intricate design of its circuit branches, featuring a multitude of components, do not constitute the most efficient approach for valve control. Future enhancements are deemed essential for

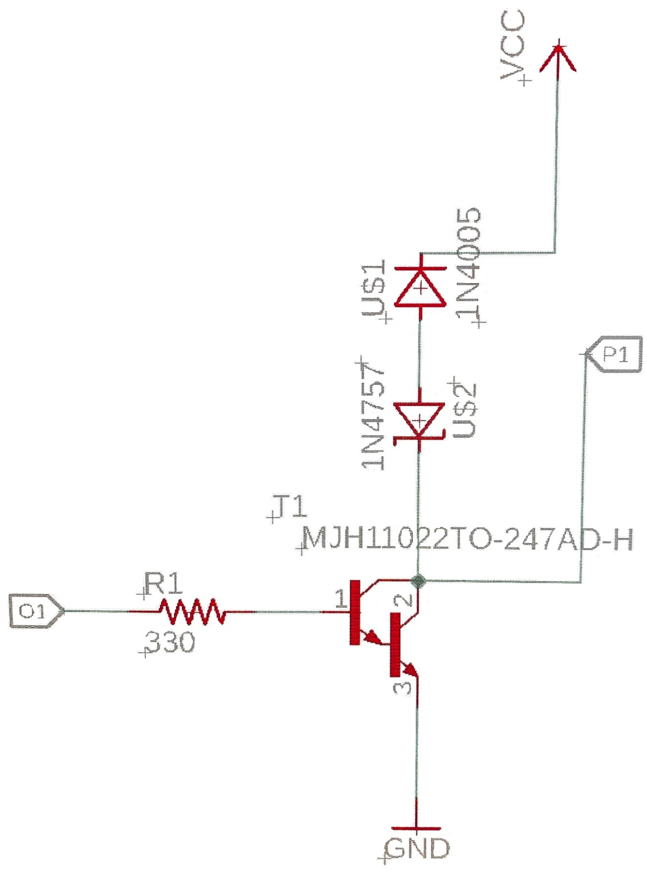


Figure 4.2: The circuit of a branch in the first-generation circuit board shown in Fig. 4.3.

several reasons. Firstly, the circuit board's size, being nearly 6 inches by 12 inches, limits it to only 13 output ports. This limitation presents a challenge for projects aiming to control hundreds or thousands of valves, leading to inefficient space utilization. Secondly, the complex assembly of components on the board is not justified by the requirements of this experiment, which does not necessitate high signal precision. Given the wide voltage range for toggling between states and the permissible level of signal noise, the employment of transistors and diodes may not represent the optimal strategy.

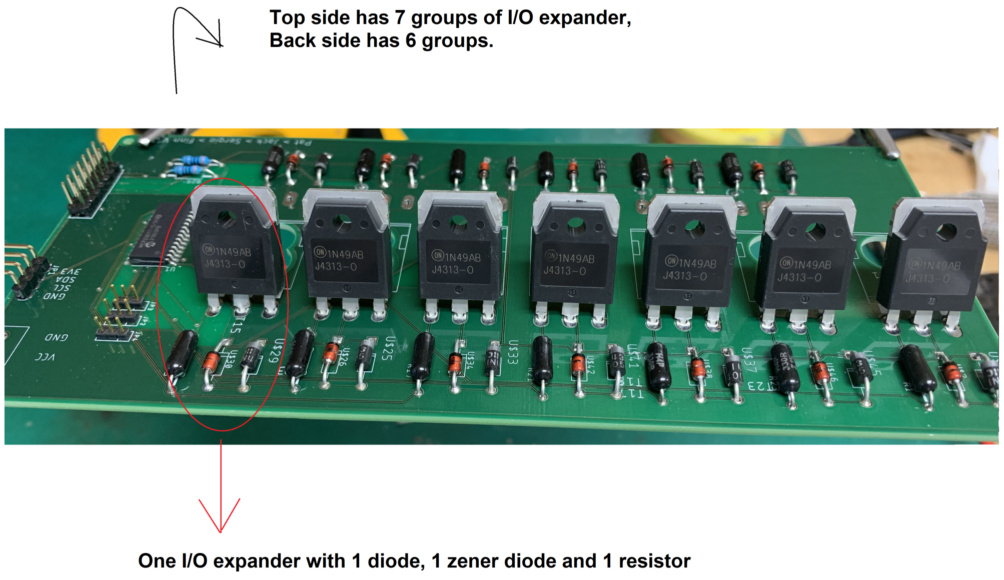

Top side has 7 groups of I/O expander,
Back side has 6 groups.
One I/O expander with 1 diode, 1 zener diode and 1 resistor

Figure 4.3: Optical photo of the first-generation circuit board.

Future modifications will aim to preserve the benefits identified in the initial design, particularly the utilization of the I^{2}C protocol, while addressing its drawbacks in terms of size and component complexity. The goal is to refine the design to overcome these limitations, focusing on enhancing both the board's compactness and its functional efficiency.

4.4 Using Adafruit as the second-generation circuit board

Given the continued use of the I^{2}C protocol, the initial approach to the circuit board's redesign leaned towards a custom solution. However, the market offers a variety of semiconductor chips that support the I^{2}C protocol. A standout option for our project became the Adafruit 16-Channel 12-bit PWM/Servo Driver with an I^{2}C interface, model PCA9685 [36]. This board is capable of managing up to 16 servo motors individually and features an I^{2}C-controlled PWM driver with an onboard clock, making it highly suitable for our needs. It is 5V compliant, allowing operation from a 3.3V microcontroller and 3.3V logic pull-ups, while still supporting output voltages up to 5.5V. With 16 address selection pins, the possibility exists to chain up to 62 of these boards together on a single I^{2}C bus, potentially providing up to 992 outputs in total. Key advantages of this board include a 1.6 KHz frequency and a 12-bit resolution for each output, which translates to roughly 4 µs resolution at a 60 Hz update rate, along with configurable push-pull or open-drain output and an output enable pin that allows for the quick deactivation of all outputs.

Figure 4.4(a) displays the Adafruit circuit board in its original state as provided by the manufacturer, while Figure 4.4(b) illustrates the board after attaching the necessary connectors and headers for our experiments. This newer generation board is markedly more user-friendly in terms of assembly compared to its predecessor, demanding significantly less soldering. Additionally, it boasts a more compact design and offers an increased number of PWM control outputs. The circuit schematic for the Adafruit board, detailing these improvements, is presented at the conclusion of this chapter in Figure 4.10. Discussion on the implementation of software control utilizing the I^2C protocol will be deferred to the subsequent chapter.

4.5 Using NXP OM13541 as the second-generation circuit board

The PCAL6534, an ultra-low-voltage, 34-bit Fm+ I^2C-bus/SMBus I/O expander, presents a viable option for its capacity to provide 34 output pins through a single chip. However, the task of manually soldering such a diminutive chip, measuring only 5 millimeters, onto a board is highly challenging. As a result, it became imperative to evaluate different manufacturers' implementations of the PCAL6534, with the NXP OM13541 emerging as the leading candidate. Figure 4.5 illustrates the NXP OM13541 circuit board.

This board is significantly more compact than its first-generation counterpart, occupying just about one-third of the size yet offering nearly triple the number of output ports. It also supports a broad spectrum of voltage levels and I^2C addressing options. The integration of testing features, as well as pull-up resistors and capacitors, enhances both the functionality and safety of the setup. Performance-wise, the NXP OM13541 represents a considerable improvement over its predecessor. The schematics for the OM13541 are shown at the end of this chapter in Figure 4.11.

With 34 digital outputs on the NXP board, optimizing the use of each pin is crucial. In an autoculture scenario with 25 cells, where each cell requires one input and one output tube, 25 pins are allocated for these wells. This leaves nine digital signals available for other uses. If these nine signals are configured as selection pins, they can generate 512 unique configurations, ranging from 000000000 to 111111111, allowing for digital control of 512 sets of tubing or 256 wells in total.

The interaction between the NXP board and a Raspberry Pi uses the I^2C protocol [37]. The initial step involves setting up the I^2C connection, with the Raspberry Pi as the primary (command issuer) and the NXP as the secondary (receiver). To activate I^2C on the Raspberry Pi, one must access the I^2C settings via "sudo raspi-config" and enable the I^2C protocol. Following this, it's necessary to install I^2C protocol utilities, for example, using "sudo apt-get install I^2C-tools." The "I^2Cdetect" command can then be used to check the address occupancy on the Raspberry Pi, with the interface for I^2Cdetect displayed in Fig. 4.6.

4.6 Servo (servomechanism): a new motor replacement

In our initial experiments utilizing the first-generation circuit board, solenoid valves were employed for the injection of culture fluid. However, given the cost considerations, servo motors emerge as a more cost-effective and dependable option compared to the pricier solenoids. A servo motor is a straightforward device that interprets digital signals as PWM (Pulse-Width Modulation), effectively mimicking an analog signal. The motor's rotation angle is determined by the duration of a square wave signal within a specific range, typically allowing for a maximum rotation of 180 degrees. The exact angle of rotation corresponds to the duration of high voltage within this range. For example, a high voltage applied for 0.25 seconds followed by a low voltage for 0.75 seconds in a one-second period (25% duty cycle) results in a rotation of 45 degrees, which is a quarter of the full 180 degrees, thereby simulating an analog signal effect.

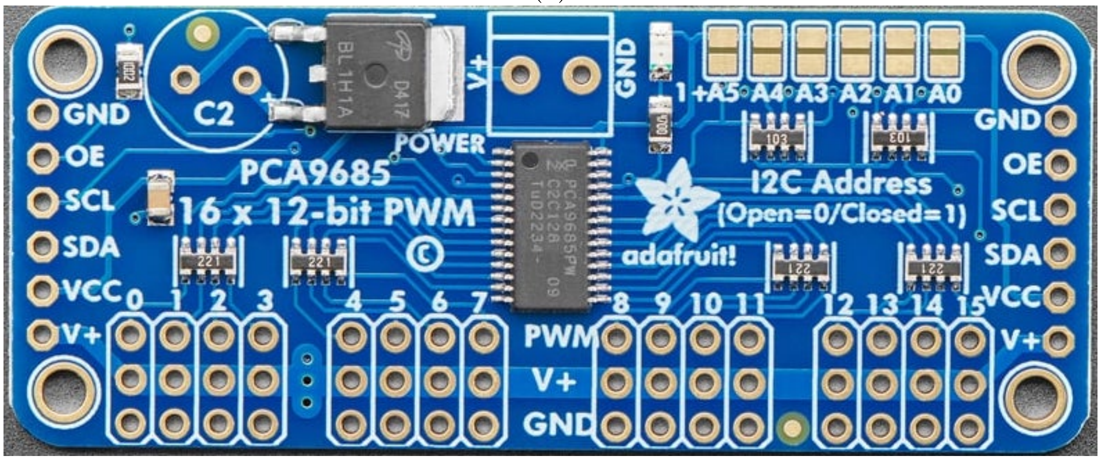
Pulse Width Modulation (PWM) represents a method of encoding a digital signal, widely utilized in diverse applications, including advanced control systems. In this context, it serves to regulate the position of a servo motor by varying the duration of time the signal remains high within a given cycle. Despite the binary nature of the signal—being either high (typically 5V) or low (ground)—adjusting the high signal's duration relative to the low phase allows for analog-like control over consistent time intervals.

The period during which the signal remains high is referred to as the "on time." The concept of the duty cycle quantifies this on time as a percentage of the total cycle time. For instance, if a digital signal is active (high) for half of its cycle and inactive (low) for the other half, its duty cycle is described as 50%, creating an ideal square wave pattern. Should the duty cycle exceed 50%, it indicates that the signal maintains a high state for a majority of its cycle. Conversely, a duty cycle of less than 50% suggests that the signal spends more time in the low state. Figure 4.7 visually represents these scenarios, showing the variations in duty cycle and their impact on the signal's high and low states.

A standard servo motor is designed to receive updates at a frequency of every 20 milliseconds, with the pulse width varying between 1 ms and 2 ms. This corresponds to a duty cycle ranging from 5 to 10% on a 50 Hz waveform. When the pulse width is set to 1.5 ms, the servo aligns to its neutral position at 90 degrees. A shorter pulse of 1 ms drives the servo to a 0 degree position, while a longer pulse of 2 ms positions it at 180 degrees. By adjusting the pulse width within this range, one can precisely control the servo's angle, achieving the desired position across the entire motion spectrum.

In our current experiment, the operational needs are simplified to just two positional states for the servo motor: the 0-degree and 180-degree positions. This reduces the demand for precise angle control, eliminating the requirement for the servo to adjust to various intermediate angles. Given the emphasis on cost-efficiency, the decision was made to acquire a substantial quantity of servos for subsequent utilization in valve control processes. This approach necessitated a modification in the control strategy, specifically the adaptation of software to transform standard digital signals into PWM signals for effective servo operation. The challenge then becomes how to implement PWM signal control to facilitate the straightforward actuation of valves. This requires a foundational understanding of PWM operation and how servo motors interpret these signals for accurate valve manipulation.

(a)



(b)

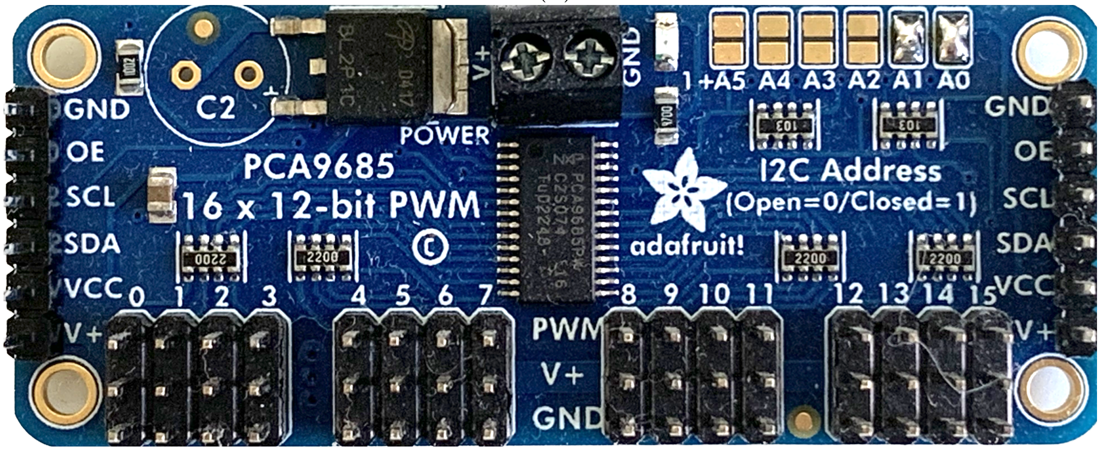


Figure 4.4: Adafruit 16-Channel 12-bit PWM/Servo Driver - I^{2}C interface - PCA9685. (a) Original board from the manufacturer. (b) Board ready for connections after soldering of various connectors.

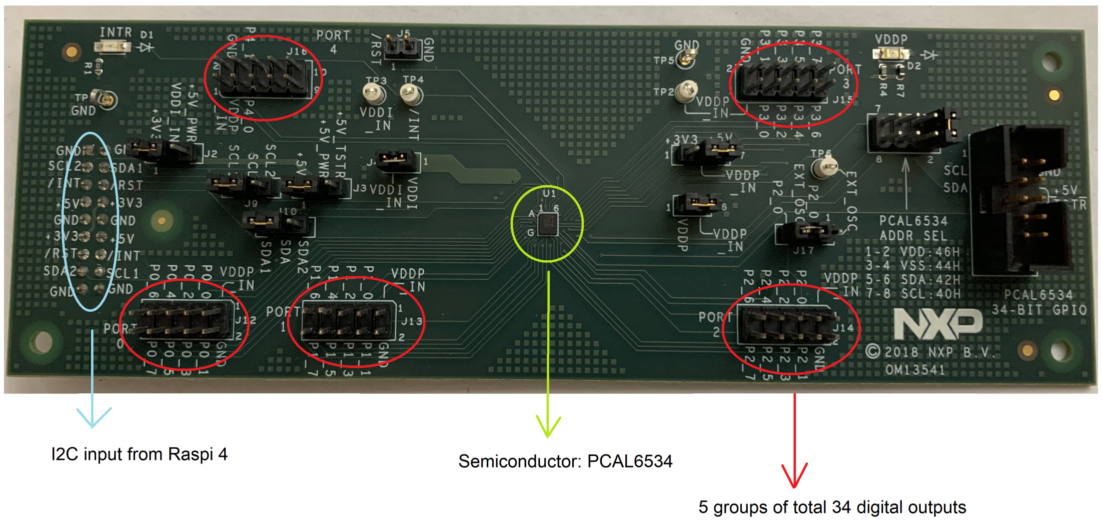
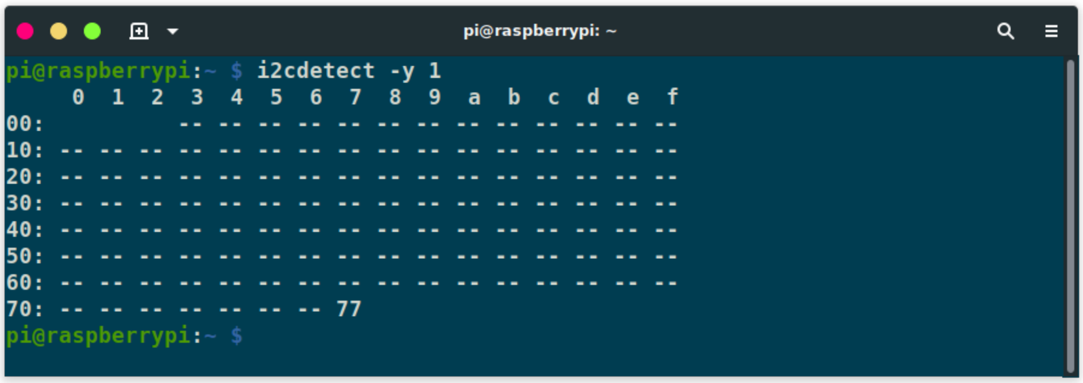


Figure 4.5: Optical photo of NXP OM13541 circuit board.



```
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- 77
pi@raspberrypi:~ $
```

Figure 4.6: I^{2}C detect interface of NXP OM13541 circuit board for monitoring the address of Raspberry Pi 4 processor.

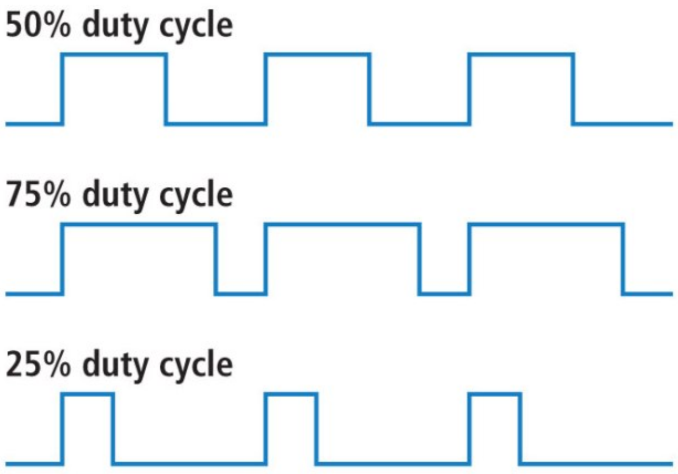


Figure 4.7: Concept of duty cycle. The duty cycle specifically describes the percentage of time a digital signal is on over an interval or period of time. The period is the inverse of the frequency of the waveform.

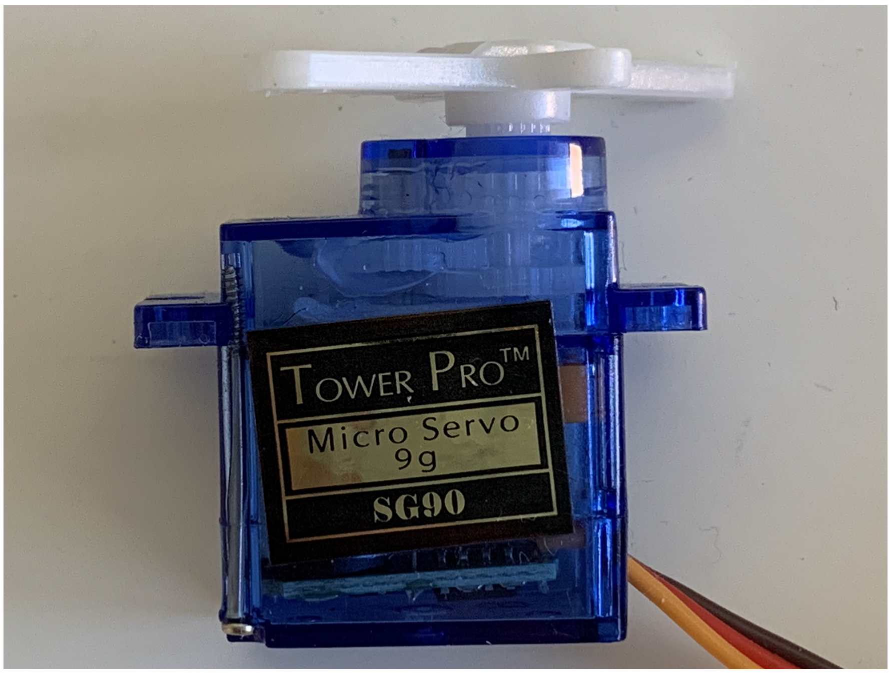


Figure 4.8: Photo of the micro servo 9g.

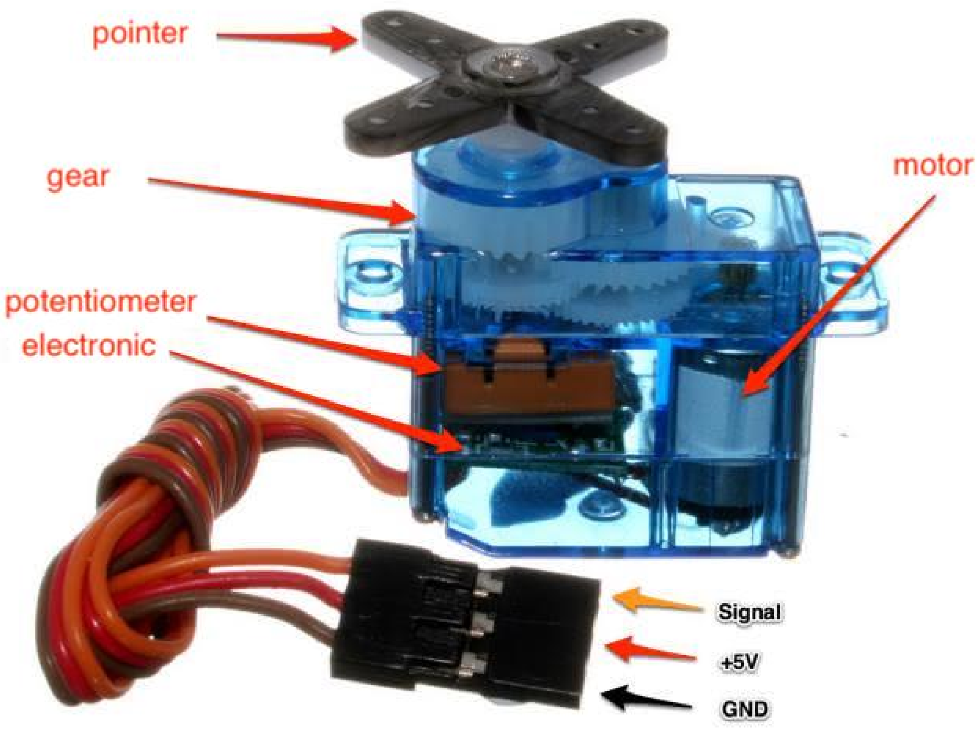


Figure 4.9: Inside view and functional parts of the micro servo 9g.

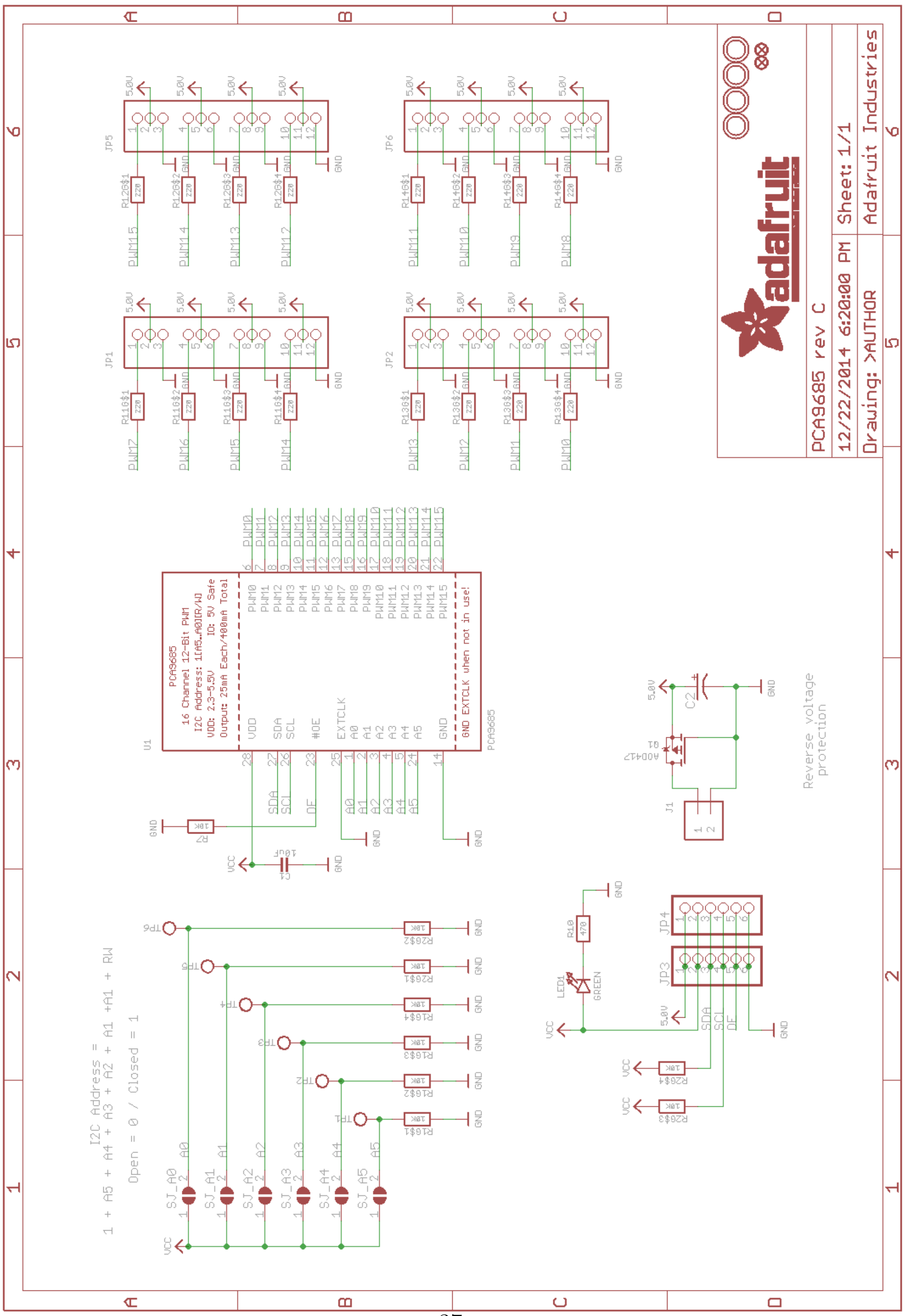


Figure 4.10: Circuit diagram schematic of Adafruit 16-Channel 12-bit PWM/Servo Driver - I^{2}C interface - PCA9685. PCA9685 is the microchip that can receive I^{2}C bus-controlled for 16 channel.

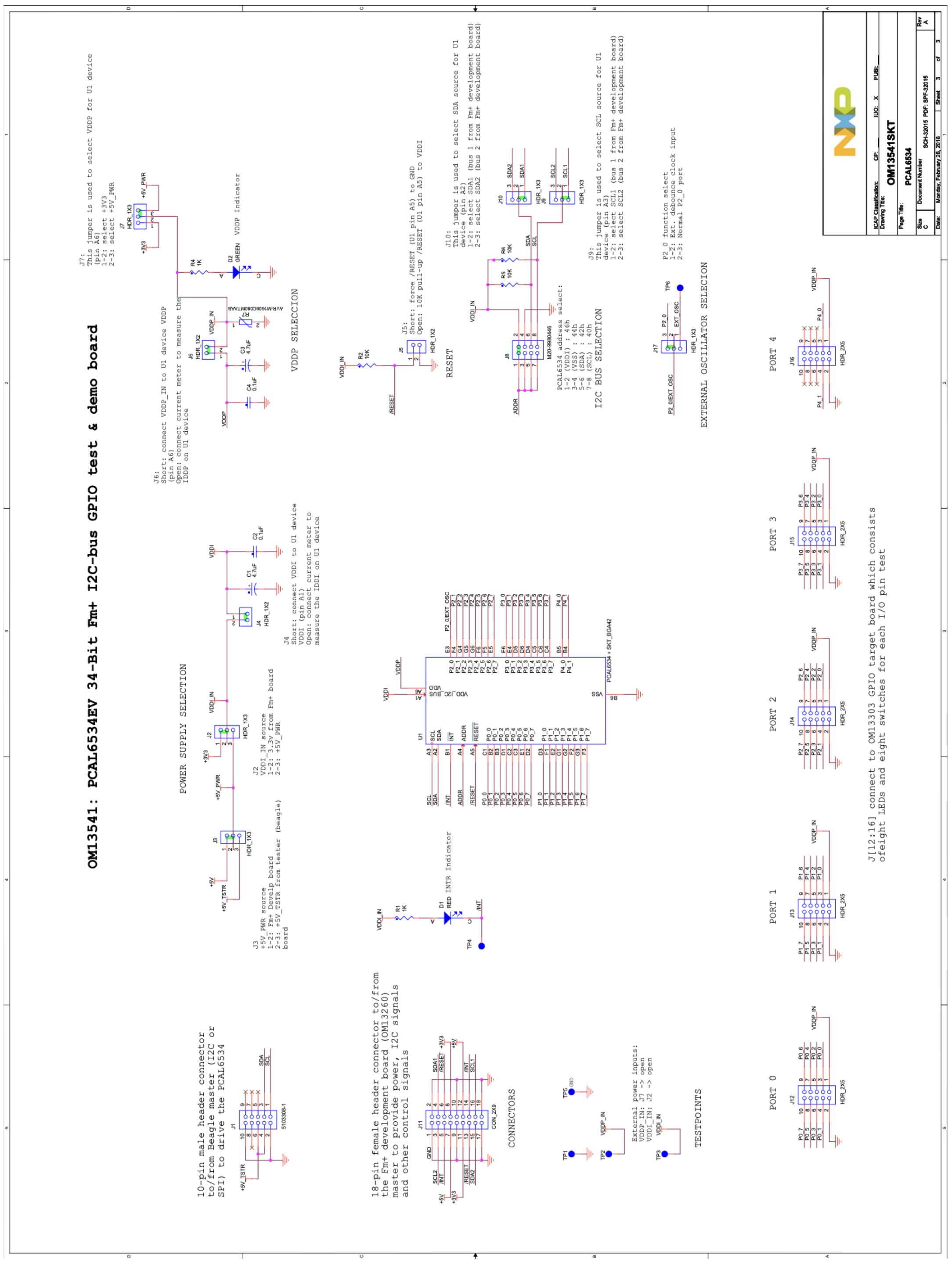


Figure 4.11: Circuit diagram schematic of NXP OM13541 circuit board.

Chapter 5

Software controls and concept

The extended-duration experiments focused on the imaging and recording of tissue cells or other biological specimens necessitate maintaining specific culture conditions over a determined period. The objective of this research is to facilitate the injection of culture fluid in precise volumes into the sample plate for designated durations and to ensure its removal at predetermined times. Consequently, this necessitates the capability to open and close valves at any desired angle and for variable lengths of time. Leveraging the hardware configuration outlined previously, achieving this level of control is possible through software manipulation of the servo motors.

5.1 Installations of operating system and software tools

To prepare the microcomputer for use, a suitable version of Linux must be loaded onto a micro SD card, which is then inserted into the Raspberry Pi 4's memory slot. The Raspberry Pi 4 is equipped with a 40-pin GPIO header on its side, which includes not only the standard GPIO pins but also pins designated for PWM, SPI, I^{2}C, and serial connections. Furthermore, the Raspberry Pi 4 is compatible with various external devices, including keyboards, mice, and monitors.

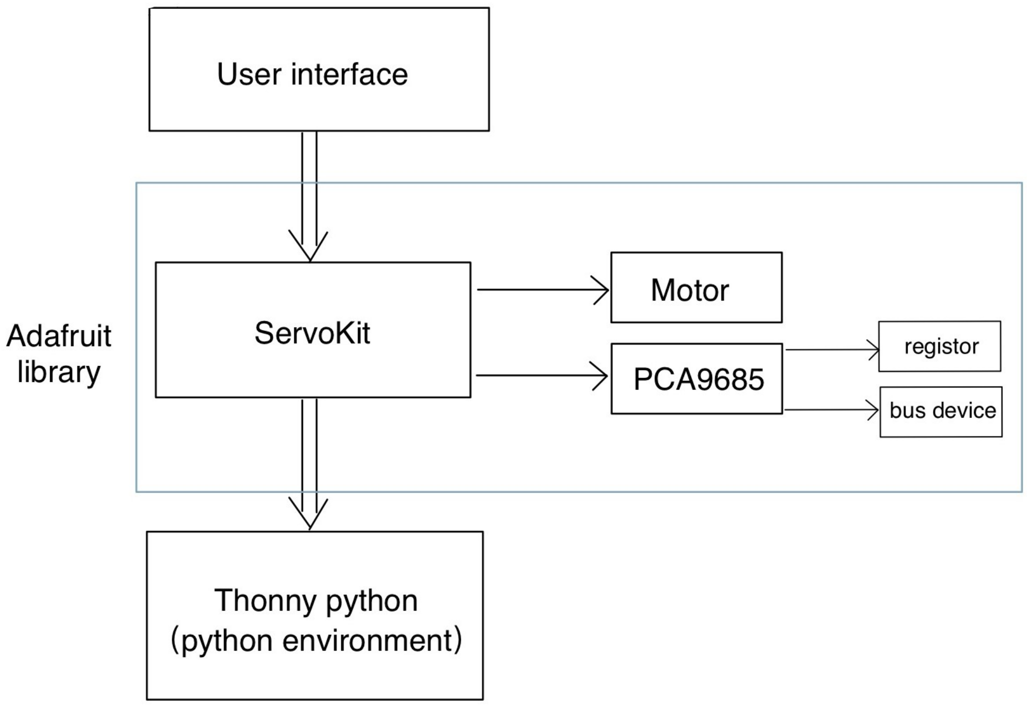


Figure 5.1: Schematic of circuit python, Adafruit library including servokit, and user interface.

Following the operating system's setup on the microcomputer, programmers typically encounter a Python development environment like 'Thonny Python' pre-installed for coding activities. Python offers access to a plethora of useful libraries available online. For coding related to the circuit board, a specific package such as 'CircuitPython' proves invaluable. This package establishes a Python-based circuit environment conducive to operating critical software tools like 'ServoKit'. ServoKit is a pivotal software library developed specifically for servo motor control by Adafruit for their PCA9685 board. This library is part of the comprehensive Adafruit Library package, as illustrated in Fig. 5.1.

Using ServoKit simplifies the programming process for servo motor manipulation. It allows the programmer to specify a sequence of pulse signals with the necessary duty cycle by simply setting an angle in the Python code, corresponding directly to the desired angle of the servo motor or liquid valve. In the forthcoming chapter, we will demonstrate how Python programming on the microcomputer can generate the required pulse signals with the appropriate duty cycle via the Adafruit board's output channel.

5.2 I^{2}C protocol and its expansion

This section provides a concise overview of the I^{2}C (Inter-Integrated Circuit) protocol fundamentals. The I^{2}C communication protocol utilizes four essential connections: the ground wire, the power supply wire, the SDA (Serial Data) line, and the SCL (Serial Clock) line.

In the I^{2}C framework, data transmission occurs through structured messages, which are segmented into data frames. Each message commences with an address frame that specifies the binary address of the device being communicated with (the slave device), followed by one or more data frames that carry the actual data to be transmitted.

Furthermore, messages are delineated by start and stop conditions, and incorporate read/write indicators, as well as ACK (acknowledge) or NACK (not acknowledge) signals after each data frame, as depicted in Fig. 5.2 [38].

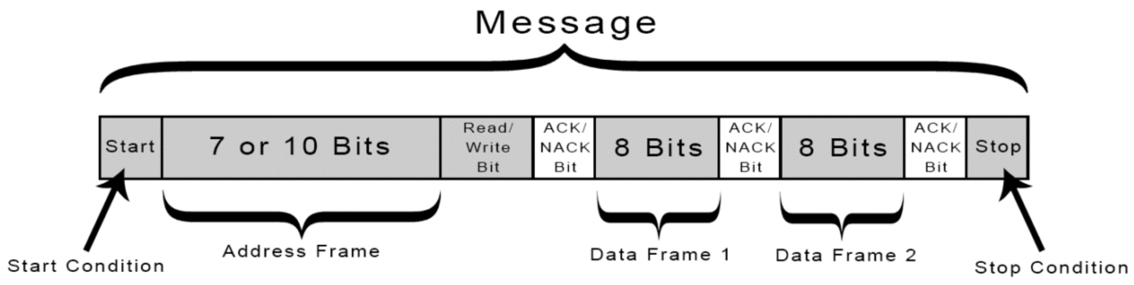


Figure 5.2: I^{2}C start stop.

Moreover, the I^{2}C protocol's ability to address multiple devices allows for the simultaneous communication with various circuit boards, each assigned a unique address. This enables a single master device to interact with numerous slave devices. Data transfer commences when the intended slave device is identified within the Address Frame, a feature that is particularly advantageous for this experiment's design, given the substantial number of samples (digital outputs) it encompasses. This attribute of I^{2}C not only meets the current experimental needs but also offers ample scope for scaling up in future projects.

The addressing feature of I^{2}C facilitates the management of several slave devices from one master device. With the standard 7-bit addressing, it is possible to have 128 (2^7) distinct addresses. While 10-bit addressing is less common, it expands the potential addresses to 1,024 (2^{10}). The configuration for connecting multiple slave devices to a single master, including 4.7K Ohm pull-up resistors linked to the SDA and SCL lines and the power supply (Vcc), is illustrated in Fig. 5.3 [38].

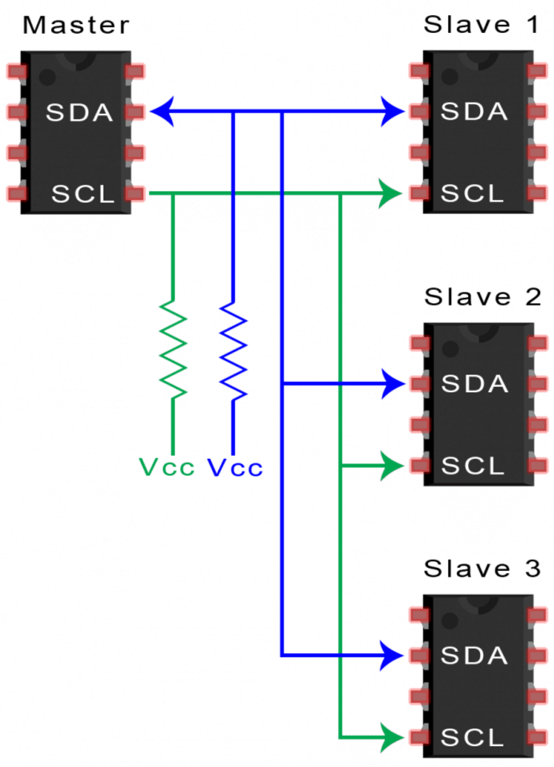


Figure 5.3: I^2C wiring with multiple secondaries.

Pull-up resistors are implemented to maintain a defined voltage level across critical components, ensuring that a wire defaults to a high logical state in the absence of an input signal. Conversely, pull-down resistors are employed to anchor inputs to logic systems at expected low logic levels when external devices are not connected or present high impedance. This setup guarantees that the wire maintains a low logic

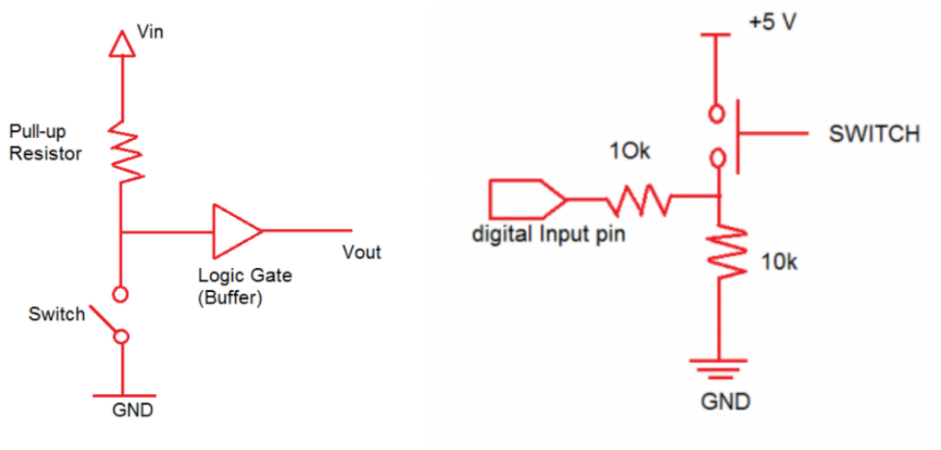


Figure 5.4: Circuit diagrams for pull-up(left) and pull-down(right) resistors.

state without active device connections, anchoring the input voltage to ground to avoid undefined input states. The resistance of a pull-down resistor should exceed the logic circuit's impedance, as depicted in Fig. 5.4 [39].

5.3 PWM operation

Beyond simply varying duty cycles to emulate different voltage levels, managing Pulse Width Modulation (PWM) intricately involves synchronizing the built-in clock with the servo's operational cycle. Essentially, this entails dictating precisely when a servo should begin processing a PWM signal segment and ensuring that the timing of signal transmission and reception aligns perfectly within each cycle.

This concept is known in telecommunications as "Signaling," highlighting the critical role of the clock counter. The clock counter ensures the synchronization of the internal clock with the device, maintaining uniformity across cycles. Although the counter primarily ensures signal alignment—as cycles typically remain constant and seldom alter—the frequency of the clock dictates the rate at which the counter adjusts.

Figure 5.5 visualizes the synchronization process between the counter and the clock, akin to ensuring two runners start and end a race at the exact same moments, where the commencement and conclusion of each signal segment must align flawlessly. In addressing this challenge, a variety of libraries and tools within Python modules have been developed. The specific coding requirements may vary depending on the experimental setup and the I^{2}C protocol's application, as each experiment's objectives and needs can differ significantly.

5.4 Further notes of the servo and Adafruit's layout

Leveraging new libraries and software tools significantly simplifies the utilization of the "Adafruit 16-Channel 12-bit PWM/Servo Driver - I^{2}C interface - PCA9685" circuit board. This board, incorporating the PCA9685 chip, is adept at translating I^{2}C signals into 16 separate digital outputs, which are subsequently modulated into PWM signals for each output. The process involves employing Python modules to synchronize the PWM signal's output with its reception at the software layer, as previously discussed.

Each servo motor connects via three wires: one for ground, one for power (voltage supply), and one for the PWM signal. The PWM signal, a square wave, manipulates the servo's rotation from 0 to 180 degrees by varying the voltage from zero to its maximum (5V in this scenario). The servo acts as a valve, controlling the flow or cessation of media through the wells. The Adafruit board acts as an intermediary, accepting I^2C signals from the Raspberry Pi and generating the requisite PWM signals. It utilizes the PCA9685 chip to accomplish this task.

For the current experiment, which involves the operation of 50 servo motors, a total of 50 distinct signals are required to manage 50 valves. These signals are dispatched through the I^2C protocol, employing four Adafruit boards and a single Raspberry Pi, culminating in 16 wires. Since each servo is connected to an individual PWM output port, an additional 50 wires are needed, considering that the manufacturer combines the ground, power, and signal wires of the servo into one. The configuration of this setup is detailed in Figs. 5.6.

5.5 Ready for experimental measurement

Figure 5.7 depicts a Raspberry Pi 4 microcomputer (the primary device) linked in series to multiple Adafruit PCA 9685 boards (the secondary devices). The objective is as follows: through Python programming on the microcomputer, it is possible to dispatch pulse signals from the jth output channel of the ith circuit board at specified times and for predetermined durations.

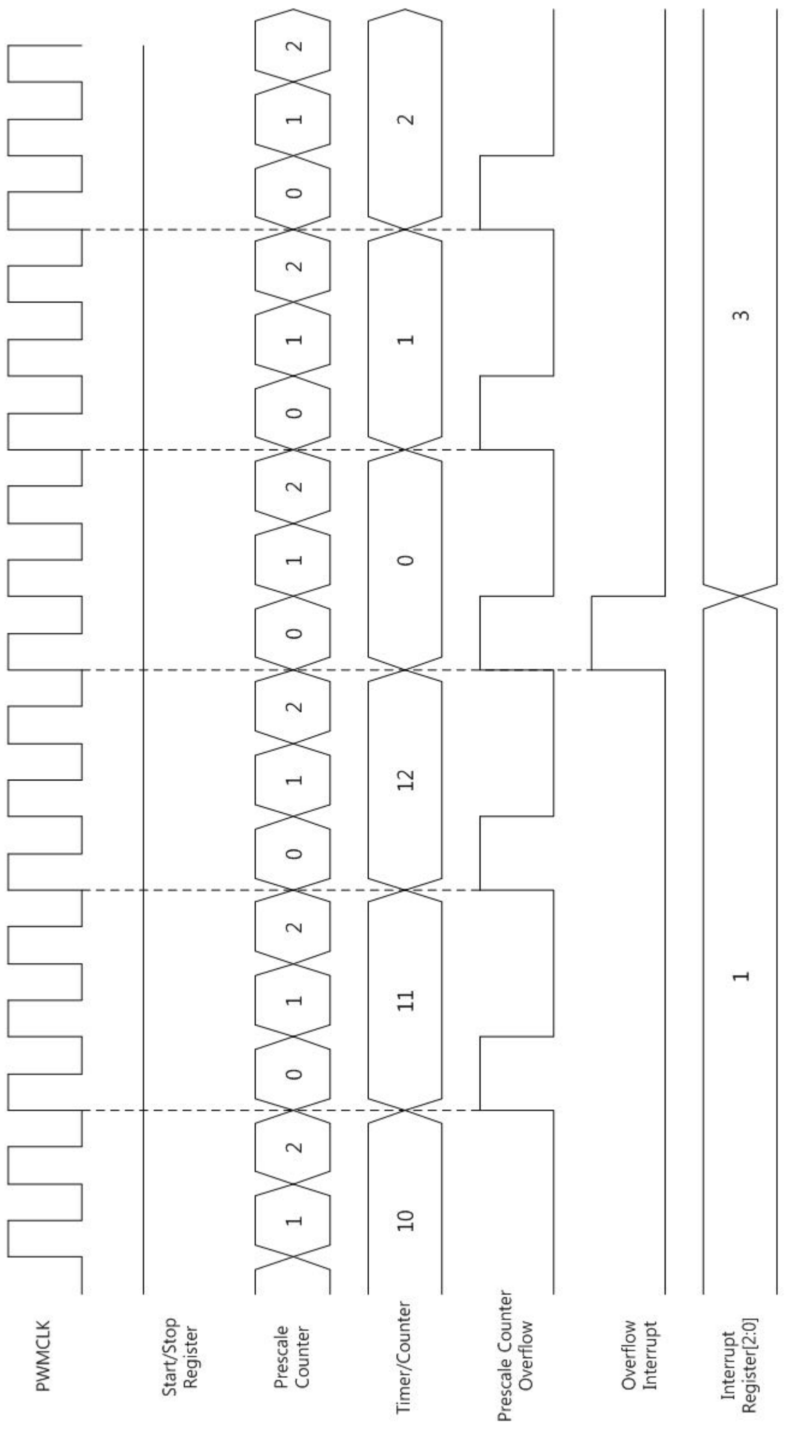


Figure 5.5: Schematics of aligning the counter and clock for timing in PWM operation.

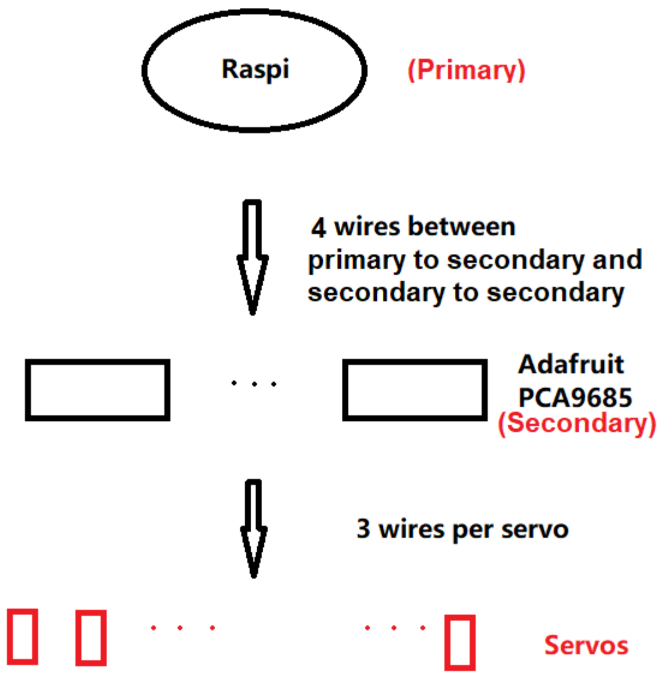


Figure 5.6: Number of wires for connection between Raspberry Pi 4 (primary) and Adafruit PCA 9685 (secondary), and between Adafruit PCA 9685 and servo motors.

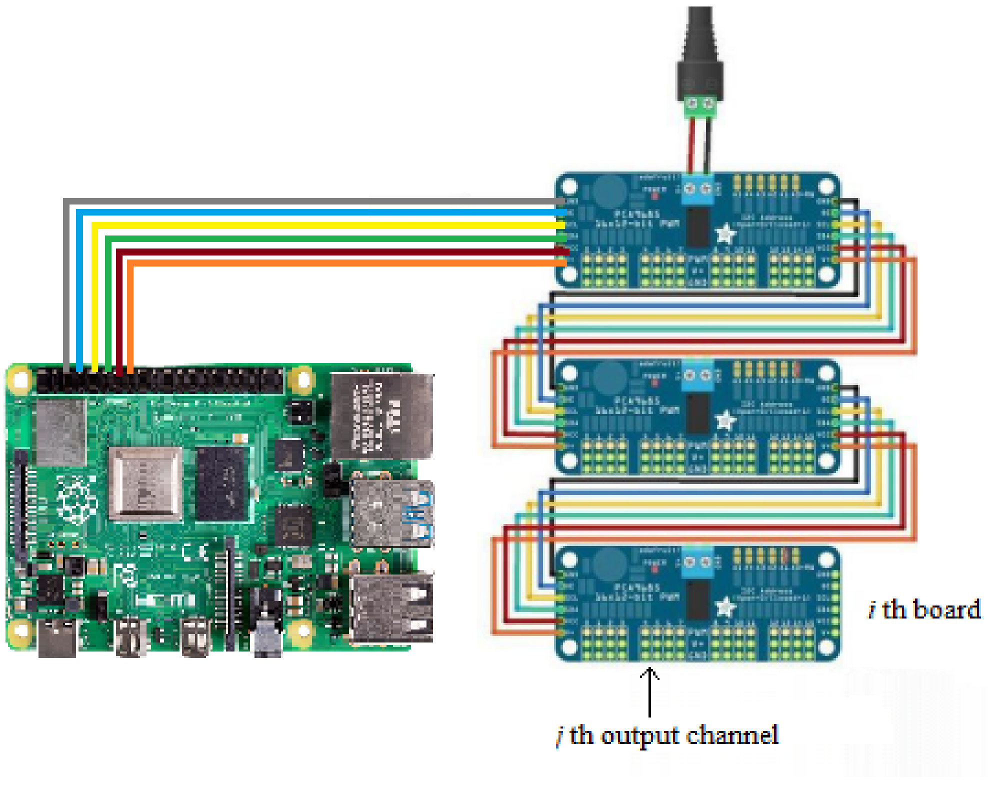


Figure 5.7: One Raspberry Pi 4 microcomputer (primary) connected to many Adafruit PCA 9685 circuit boards in series (secondaries). Our goal is like this: By python programming on the microcomputer, one is able to send out pulse signals from the jth output channel of the ith circuit board at given times for given time intervals.

Chapter 6

Results and discussion

Following the assembly of the hardware components and the setup of the software controls, we have set the stage to showcase the functionality of the liquid valves that facilitate the injection and extraction of biological culture fluid. Building on the servo motor's capabilities described earlier, our focus now shifts to demonstrating how Python programming on the Raspberry Pi 4 microcomputer can generate and dispatch pulse signals with arbitrary duty cycles and a period of 20 ms from the Adafruit circuit board's output pins. These signals can be precisely timed and sustained for specific durations.

6.1 PWM control

In the current study, the micro servo 9g (model SG90), as depicted in Fig. 4.8, it was observed that its 180-degree rotation span corresponds to PWM pulses over a period of 20 ms with pulse widths ranging from 0.5 ms to 2.5 ms. Moreover, pulse widths of 0.5 ms and 2.5 ms, which correspond to 0 and 180 degrees, respectively, can be generated from the Adafruit board's output pins using Python programming through the ServoKit package to set the angles.

Oscilloscope captures showcasing the PWM pulses with a 0.5 ms pulse width for a 0-degree setting are presented in Fig. 6.1(a) displays a series of pulses over a 20 ms period, while Fig. 6.1(b) explicitly shows the 0.5 ms pulse width. Similarly, oscilloscope images for pulse widths of 1.5 ms and 2.5 ms, corresponding to 90 and 180 degrees, respectively, are depicted in Fig. 6.2 and Fig. 6.3.

The Python scripts for servo motor control are documented in Appendix **??**. A video file named "servo.mp4" includes a visual demonstration of the rotations to these three positions, activated via the Python script. This illustrates that rotation to any angle within the 0 to 180-degree range is feasible by specifying the desired angle in the Python program.

6.2 Long term performance

The clarity and precision with which the PWM signal is transmitted to the Servo are evident. The Servo has the capability to accurately control angles of 90 degrees or 180 degrees, facilitating precise adjustments in the solution's flow. The effectiveness of a single Servo in executing this task suggests that a system of multiple Servos would operate in a correspondingly effective manner. The focus then shifts to assessing long-term control capabilities.

During a 6-hour experimental run, the stability of the signal was monitored, revealing no deviations in the Servo's performance from start to finish. This consistency underscores the reliable toggling between on and off states. One observation was a slight increase in the CPU temperature of the Raspberry Pi. To mitigate this in future experiments, maintaining a stable ambient temperature and implementing cooling strategies for the CPU are recommended. Overall, it was determined that long-term operation of the circuit is not adversely impacted by extended use.

6.3 Multiple servo performance

Following the verification of hardware performance, the remaining step involves finalizing the software for operating the 50 valves. Subsequently, using the ServoKit library, the servos' rotation angles can be controlled. This can be achieved by defining a class, potentially named "braingeneers-servo," which allows for the manipulation of any servo across the four Adafruit boards to rotate to either 0 or 180 degrees. Within this class, functions dedicated to the opening and closing actions can be implemented to dictate the servo's movements. In the primary function, it is essential to ensure that the user's input falls within the range of 0 to 49, each number corresponding to a specific servo.

6.4 Results

The initial evaluation was aimed at determining the accuracy of the servo mechanisms' responses. To achieve this, four servos underwent a series of activation and deactivation tests. As illustrated in Figure 6.4, the response times for each of the four servos were precisely measured, revealing that the reaction times were impressively concise, spanning only 10 milliseconds and specifically clocking in at 2, 4, 5, and 6 milliseconds, respectively. Such results clearly fall within the defined tolerance limits, demonstrating satisfactory performance.

The next phase of assessment focused on the collective behavior of multiple servos when activated simultaneously by a single signal from a Raspberry Pi. Figure 6.5 showcases the targeted activation (on) and deactivation (off) sequences, scheduled to alternate every second for a duration of 15 seconds. Further analysis, as depicted in Figure 6.6, breaks down the servo operations into three phases: the first phase exhibits immediate response without delay, the second phase shows a slight delay of 0.1 seconds, and the third phase indicates a more noticeable delay of 0.2 seconds. This setup involved arranging four servos in the initial phase and three servos in each subsequent phase, totaling ten servos. According to the findings recorded in Tables 6.1 and 6.2, all servos adhered to the response time criterion, maintaining performance within the acceptable 10-millisecond threshold.

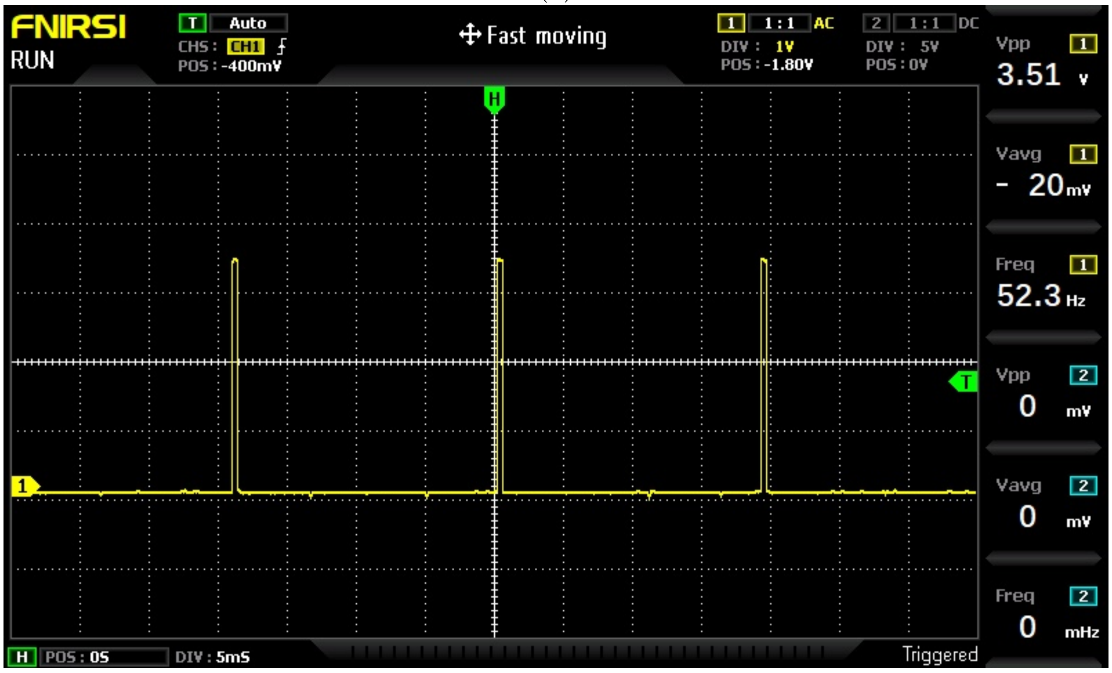
Ideal	0s (on)	1s (off)	2s (on)	3s (off)	4s (on)	5s (off)	6s (on)	7s (off)
Servo #1	0.0	1.007	2.0	3.0	4.0	5.0	6.0	7.0
Servo #2	0.0	1.0	2.0	3.002	4.0	5.0	6.004	7.001
Servo #3	0.0	1.0	2.0	3.002	4.0	5.0	6.004	7.0
Servo #4	0.0	1.0	2.001	3.0	4.0	5.0	6.004	7.001
Servo #5	0.1	1.1	2.1	3.102	4.1	5.102	6.1	7.1
Servo #6	0.1	1.1	2.105	3.102	4.1	5.097	6.1	7.099
Servo #7	0.1	1.098	2.1	3.107	4.094	5.1	6.103	7.108
Servo #8	0.2	1.2	2.2	3.201	4.195	5.202	6.198	7.201
Servo #9	0.2	1.193	2.021	3.2	4.2	5.2	6.2	7.198
Servo #10	0.2	1.2	2.204	3.2	4.2	5.201	6.2	7.201

Table 6.1: Servo responding time (0 - 7 s)

Ideal	8s (on)	9s (off)	10s (on)	11s (off)	12s (on)	13s (off)	14s (on)	15s (off)
Servo #1	7.999	9.0	10.0	11.0	12.0	13.0	14.0	15.0
Servo #2	8.0	9.0	10.001	11.0	12.0	13.0	13.998	15.0
Servo #3	8.002	9.007	10.001	11.0	12.0	13.0	13.998	15.0
Servo #4	8.0	9.004	10.004	11.001	11.996	12.991	13.998	14.998
Servo #5	8.1	9.101	10.101	11.1	12.1	13.099	14.1	15.0
Servo #6	8.1	9.107	10.099	11.1	12.1	13.1	14.098	14.999
Servo #7	8.103	9.1	10.1	11.1	12.1	13.102	14.1	15.0
Servo #8	8.202	9.2	10.2	11.2	12.2	13.2	14.2	15.0
Servo #9	8.192	9.201	10.197	11.2	12.203	13.2	14.2	14.999
Servo #10	8.2	9.2	10.2	11.2	12.2	13.2	14.205	15.0

Table 6.2: Servo responding time (8 - 15 s)

(a)



(b)

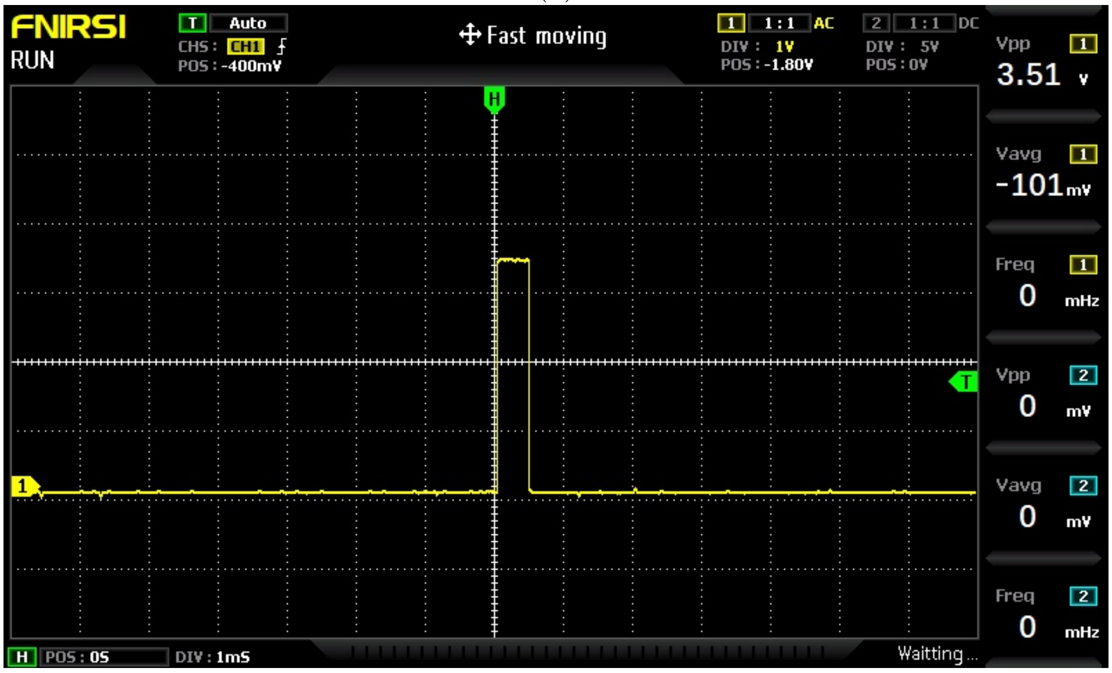
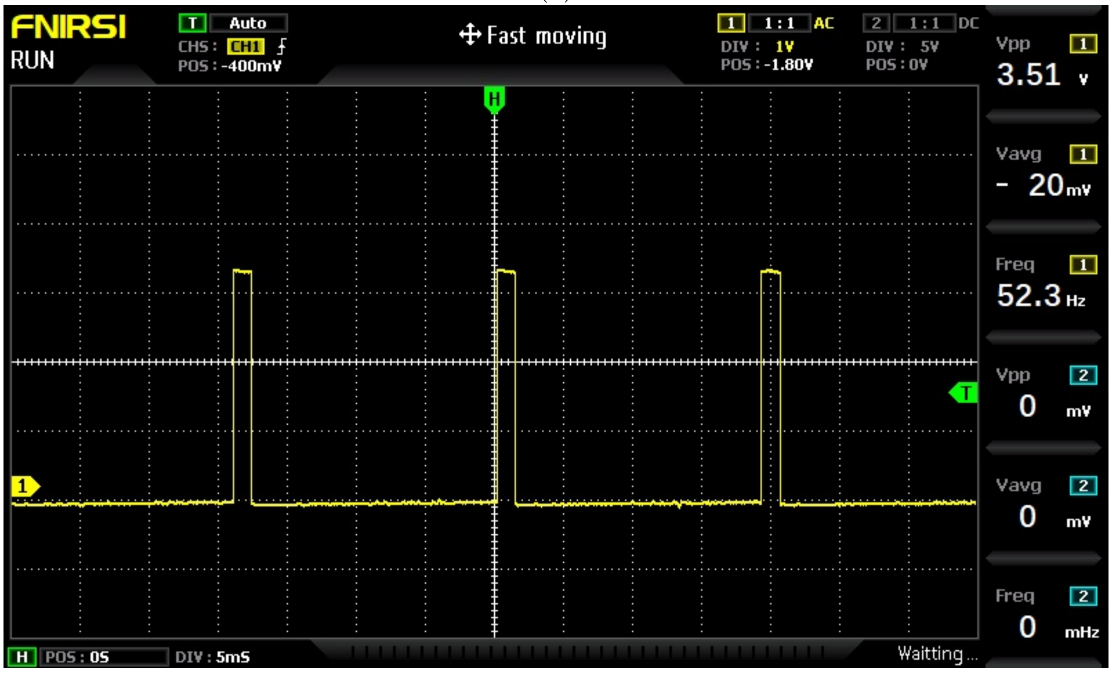


Figure 6.1: Oscillascope images of the pulse signal for driving the servo motor to an angle of 0°, produced at the output pin of the Adafruit board. All vertical axes have 1 volt per division. (a) Horizontal axis: 5 ms/division, showing a train of pulses with a period of 20 ms. (b) Horizontal axis: 1 ms/division. Pulse width of 0.5 ms can be clearly seen.

(a)



(b)

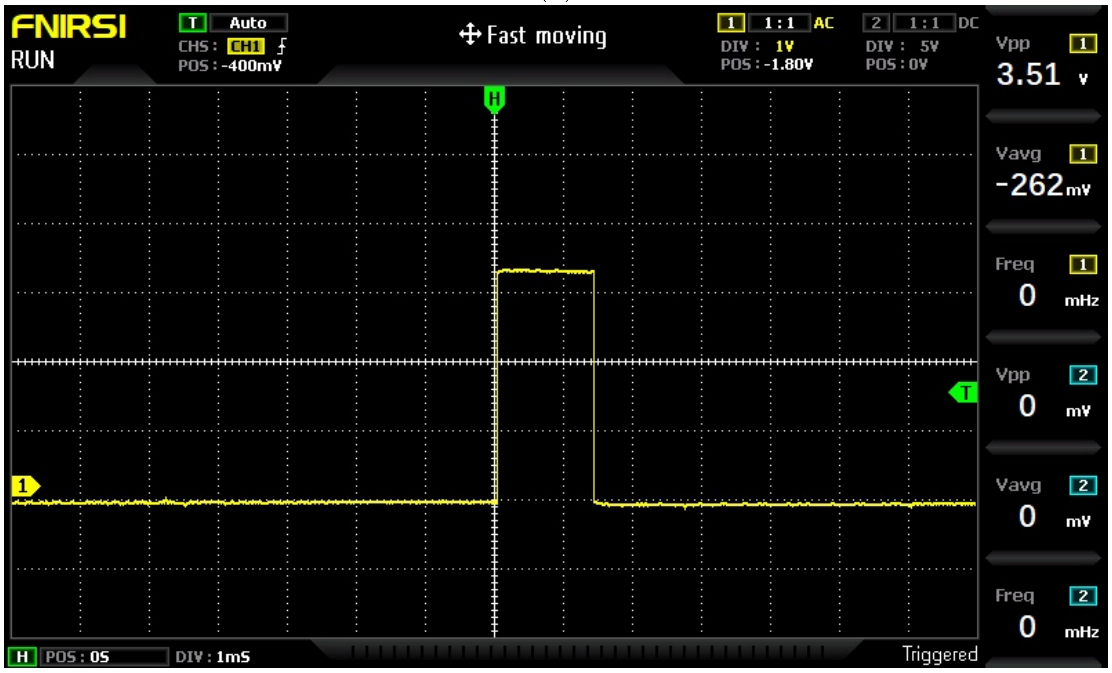
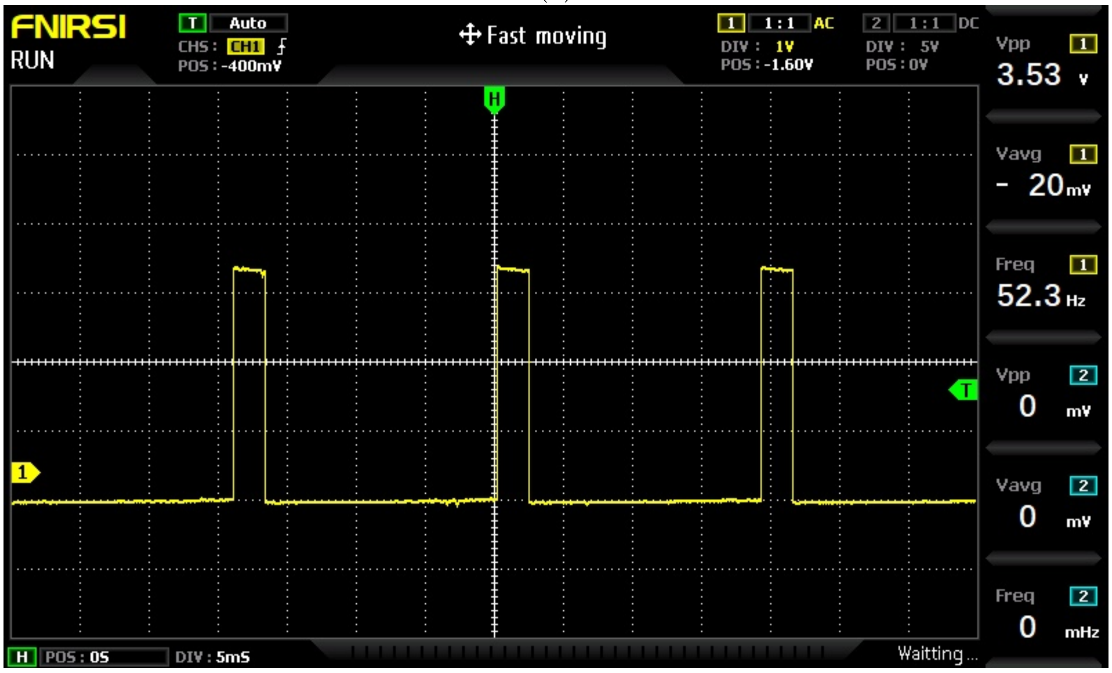


Figure 6.2: Oscillascope images of the pulse signal for driving the servo motor to an angle of 90°, produced at the output pin of the Adafruit board. All vertical axes have 1 volt per division. (a) Horizontal axis: 5 ms/division, showing a train of pulses with a period of 20 ms. (b) Horizontal axis: 1 ms/division. Pulse width of 1.5 ms can be clearly seen.

(a)



(b)

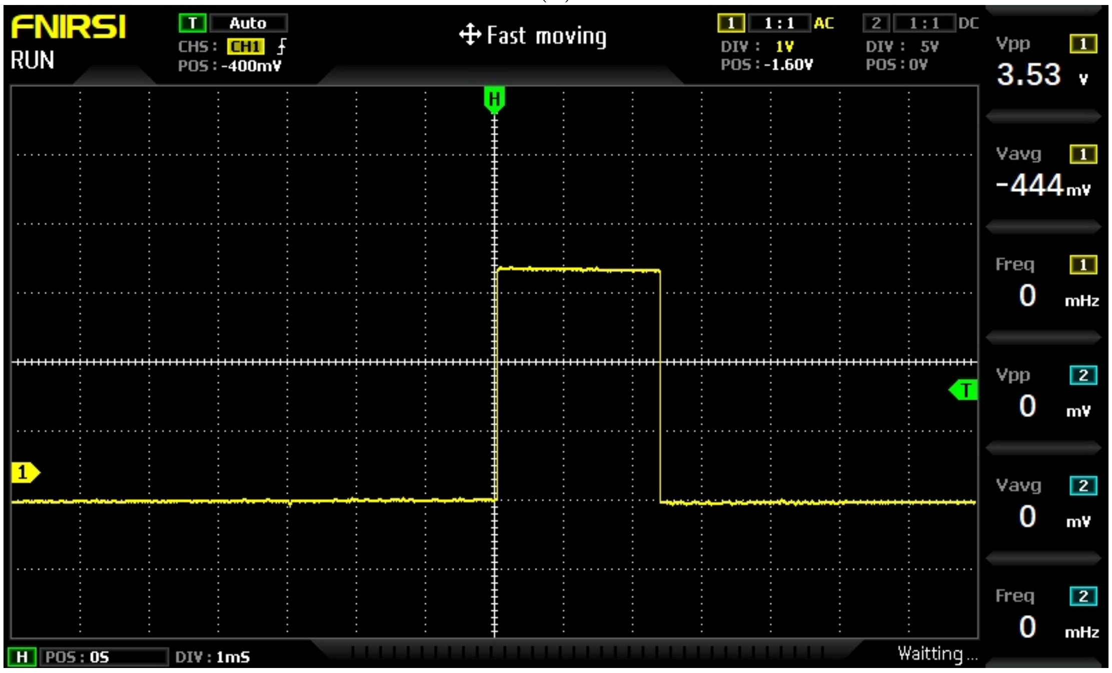


Figure 6.3: Oscillascope images of the pulse signal for driving the servo motor to an angle of 180°, produced at the output pin of the Adafruit board. All vertical axes have 1 volt per division. (a) Horizontal axis: 5 ms/division, showing a train of pulses with a period of 20 ms. (b) Horizontal axis: 1 ms/division. Pulse width of 2.5 ms can be clearly seen.

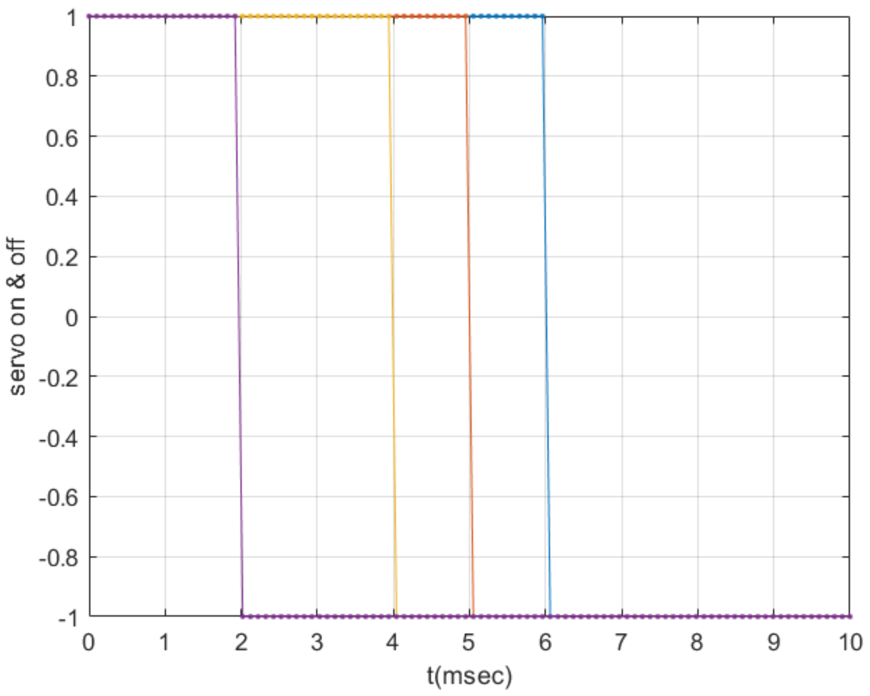


Figure 6.4: 4 Servos with millisecond of time delay

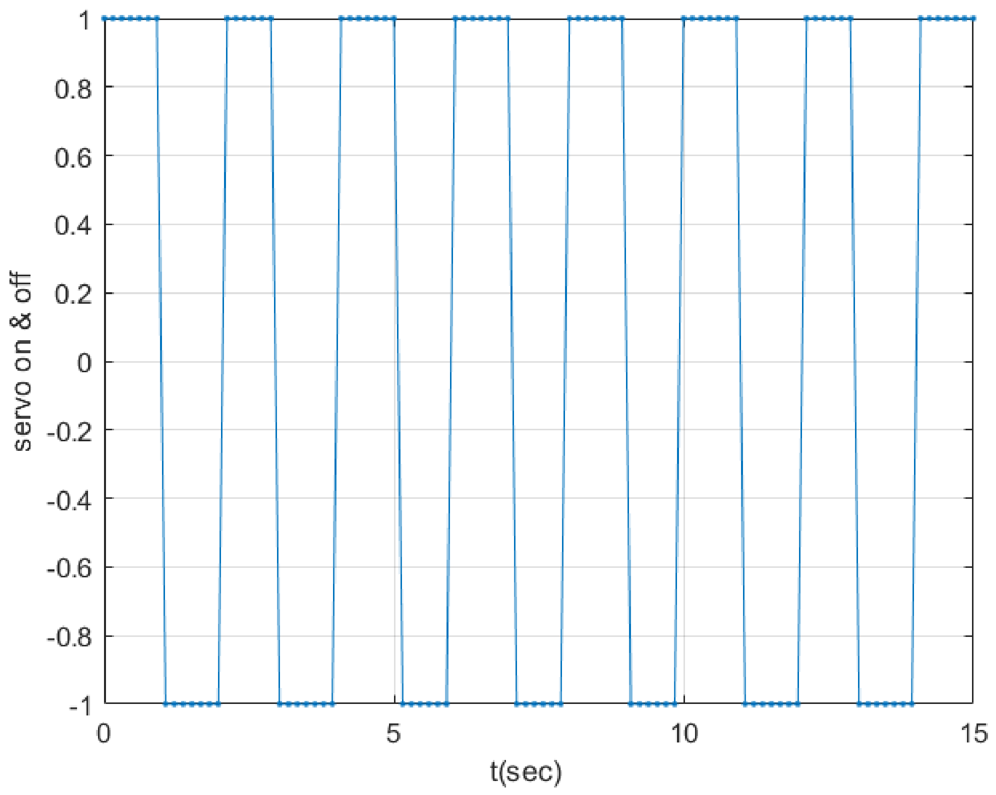


Figure 6.5: Ideal servo on and off figures plot.

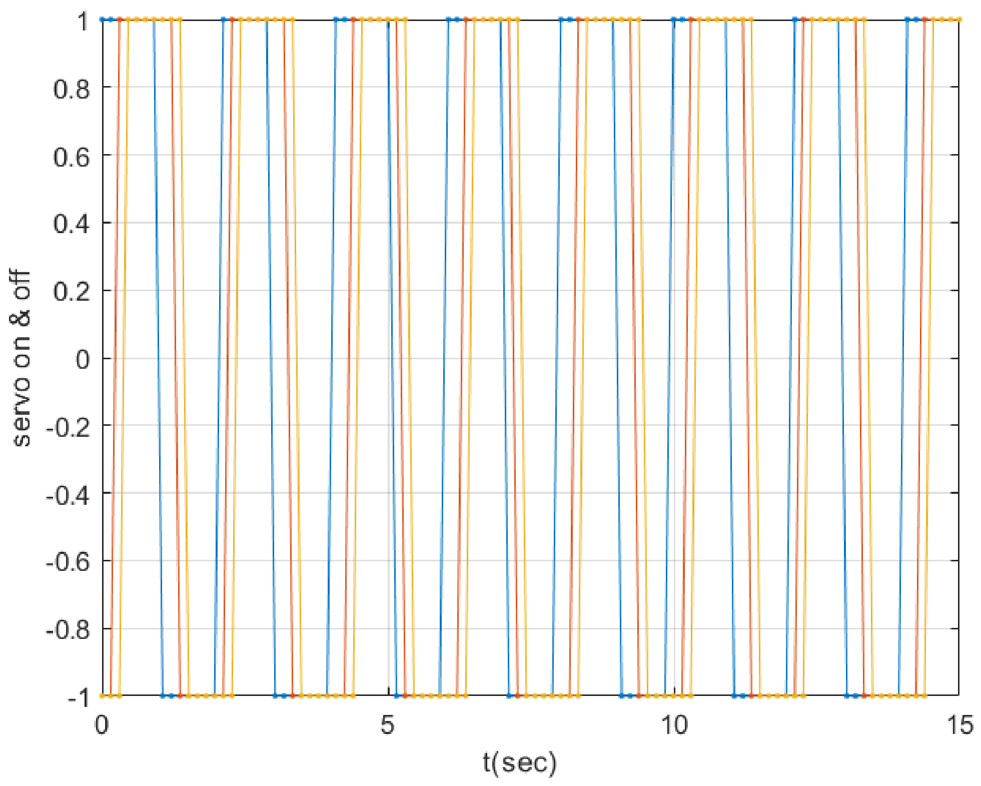


Figure 6.6: In the graphical representation under examination, the color blue is utilized to denote the optimal performance metrics, whereas the color yellow signifies the data obtained from the most different servo analysis (0.2 second slower). Concurrently, the color red is employed to depict the mean performance values, derived from an aggregate analysis of 10 distinct servos. This evaluative comparison was conducted over a temporal span of 15 seconds, serving as a testament to the detailed temporal resolution achievable in servo performance assessment.

Chapter 7

Conclusion

This thesis marks a significant achievement in the realm of laboratory automation, particularly for long-term biological research, by successfully undertaking the comprehensive task of designing, prototyping, programming, and evaluating a fully programmable valve control system. The foundation of this initiative was the creation of a valve control solution that not only offers programmability but also excels in ease of assembly, compactness, cost-efficiency, and operational reliability. Through a thorough design process, the hardware was crafted to fulfill these requirements, resulting in a product that is both diminutive and streamlined, thus ensuring its seamless integration into diverse research environments without demanding substantial space or financial resources.

On the software front, the project adopted an advanced strategy by selecting Python as the primary programming language, chosen for its wide-ranging flexibility and comprehensive support for scientific computation. The incorporation of the I^2C

protocol and the deployment of specialized software packages for hardware communication highlighted the project's dedication to facilitating flawless interaction between the control mechanism and the valve operations, enhancing the system's dependability and versatility across various experimental frameworks.

The project reached its culmination through thorough testing phases that were crucial in showcasing the valve control system's operational effectiveness. These trials not only validated the concept but also confirmed the system's ability to fulfill the stringent requirements of long-term biological experimentation. The conclusive experimental evidence affirms the system's robust performance, emphasizing its capability to transform biological research practices by offering unparalleled control and accuracy in fluid management.

Chapter 8

Future Work

Certainly, achieving the objectives of this project necessitates the completion of several precise tasks. The initial step involves creating Python software for I^2C control, which encompasses a comprehensive understanding of various elements such as the CircuitPython Library—its installation, application, and related software intricacies concerning ServoKit, including the execution of BUSio and PWMio.

The subsequent task entails the design of an intelligently structured distribution network of circuit boards. This design strategy requires the conception of a primary board alongside several auxiliary boards. Utilizing Eagle software to map out the circuit logic, the primary board is tasked with the distribution role, tasked with selecting specific target sequences from a pool of 512 binary sequences and directing these sequences to the auxiliary boards. These auxiliary boards, in turn, are designed to interpret the received sequences and establish connections with an array of solenoids or servo motors. Depending on the requirements of the experiment or application, it becomes necessary to develop output programs that can manage either straightforward digital signals or the more intricate PWM signals.

Bibliography

[1] Elizabeth A. Specht, Esther Braselmann, and Amy E. Palmer. A Critical and Comparative Review of Fluorescent Tools for Live Cell Imaging. *Annu. Rev. Physiol.* 79:11.1–11.25, 2017.

[2] T. Ly Victoria et al. Picroscope: low-cost system for simultaneous longitudinal biological imaging. *Commun. Biol.* 4:1261, 2021.

[3] T. Seiler Spencer et al. Modular automated microfluidic cell culture platform reduces glycolytic stress in cerebral cortex organoids. *Sci. Rep.* 12:20173, 2022.

[4] Kateryna Voitiuk et al. Light-weight electrophysiology hardware and software platform for cloud-based neural recording experiments. *J. Neural Eng.* 18:066004, 2021.

[5] George O. T. Merces et al. The incubot: A 3D printer-based microscope for long-term live cell imaging within a tissue culture incubator. *HardwareX* 9:e00189, 2021.

[6] G. Gürkan and K. Gürkan. Incu-Stream 1.0: An Open-Hardware Live-Cell Imaging System Based on Inverted Bright-Field Microscopy and Automated Mechanical

Scanning for Real-Time and Long-Term Imaging of Microplates in Incubator. *IEEE Access* 7:58764–58779, 2019.

[7] J. M. Pearce, Building research equipment with free, open-source hardware. *Science* 337:1303–1304, 2012.

[8] M. F. Coakley et al. The NIH 3D print exchange: a public resource for bioscientific and biomedical 3D prints. *3D Print. Addit. Manuf.* 1:137–140, 2014.

[9] S. Willis. The maker revolution. *Computer.* 51:62–65, 2018.

[10] B. Ambrose et al. Democratizing single-molecule fret: an open-source microscope for measuring precise distances and biomolecular dynamics. *Biophys. J.* 118:614a, 2020.

[11] Z. Wang et al. A high-resolution minimicroscope system for wireless realtime monitoring. *IEEE Trans. Biomed. Eng.* 65:1524–1531, 2017.

[12] J. W. Brown et al. Single-molecule detection on a portable 3d-printed microscope. *Nat. Commun.* 10:1–7, 2019.

[13] B. Diederich et al. A versatile and customizable low-cost 3d-printed open standard for microscopic imaging. *Nat. Commun.* 11:1–9, 2020.

[14] H. Zhao, C. Tang, K. Cui, B.-T. Ang, S. T. Wong. A screening platform for glioma growth and invasion using bioluminescence imaging. *J. Neurosurg.* 111:238–246, 2009.

[15] H. L. Martin et al. High-content, high-throughput screening for the identification of cytotoxic compounds based on cell morphology and cell proliferation markers. *PloS One* 9:e88338, 2014.

[16] N. Abe-Fukasawa, K. Otsuka, A. Aihara, N. Itasaki, and T. Nishino. Novel 3d liquid cell culture method for anchorage-independent cell growth, cell imaging and automated drug screening. *Sci. Rep.* 8:1–12, 2018.

[17] A. Bohm. An inexpensive system for imaging the contents of multi-well plates. *Acta Crystallogr. F: Struct. Biol. Commun.* 74:797–802, 2018.

[18] S. B. Kim et al. A mini-microscope for in situ monitoring of cells. *Lab Chip.* 12:3976–3982, 2012.

[19] S. D. Klimaj, Y. Licon Munoz, K. Del Toro, and W. C. Hines. A highthroughput imaging and quantification pipeline for the evos imaging platform. *Plos One* 15:e0236397, 2020.

[20] S. Ruckdäschel, S. Michaelis, and J. Wegener. Time lapse imaging of spheroids–zencellowl incubator. https://annualmeeting.ls2.ch/files/2021/content/364/zencellowl-incubator-microscope-time-lapse-imaging-ofspheroids.pdf (2017).

[21] G. A. Lemieux et al. A whole-organism screen identifies new regulators of fat storage. *Nat. Chem. Biol.* 7:206–213, 2011.

[22] N. Tsuji et al. Whole organism high content screening identifies stimulators of pancreatic beta-cell proliferation. *PloS One* 9:e104112, 2014.

[23] J. J. Early et al. An automated high-resolution in vivo screen in zebrafish to identify chemical regulators of myelination. *Elife* 7:e35136, 2018.

[24] Y. T. Huang, Y. L. Chang, C. C. Chen, P. Y. Lai, and C. K. Chan. Positive feedback and synchronized bursts in neuronal cultures. *PLoS One* 12:e0187276, 2017.

[25] C. A. Trujillo et al. Complex oscillatory waves emerging from cortical organoids model early human brain network development. *Cell Stem Cell* 25:558–69.e7, 2019.

[26] Y. S. Zhang et al. Multisensor-integrated organs-on-chips platform for automated and continual in situ monitoring of organoid behaviors. *Proc. Natl Acad. Sci.* 114:E2293–302, 2017.

[27] Y. Yada, T. Mita, A. Sanada, R. Yano, R. Kanzaki, D. J. Bakkum, A. Hierlemann, and H. Takahashi. Development of neural population activity toward self-organized criticality. *Neuroscience* 343:55–65, 2017.

[28] M. P. Zafeiriou, G. Bao, J. Hudson, R. Halder, A. Blenkle, M. K. Schreiber, A. Fischer, D. Schild, and W. H. Zimmermann. Developmental GABA polarity switch and neuronal plasticity in bioengineered neuronal organoids. *Nat. Commun.* 11:3791, 2020.

[29] www.wikipedia.org/wiki/Raspberry-Pi

[30] https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[31] www.raspberrypi.org/help/

[32] www.raspberrypi.org/downloads/

[33] https://www.adafruit.com/product/815

[34] https://www.nxp.com/products/interfaces/ic-spi-i3c-interface-devices/general-purpose-i-o-gpio/pcal6534-evaluation-board:OM13541

[35] https://www.sparkfun.com/servos

[36] https://learn.adafruit.com/scanning-I^{2}C-addresses/raspberry-pi —install I^{2}C tools website

[37] https://www.mathworks.com/help/supportpkg/raspberrypiio/ref/enableI^{2}C.html — enable website

[38] https://www.circuitbasics.com/basics-of-the-I^{2}C-communication-protocol/

[39] https://www.circuitbasics.com/pull-up-and-pull-down-resistors/

[40] https://www.ics.com/blog/control-raspberry-pi-gpio-pins-python

[41] C. Watson, S. Senyo. All-in-one automated microfluidics control system. *ScienceDirect* 5:e00063, 2019.

[42] B. Li, et al. A smartphone controlled handheld microfluidic liquid handling system. *LabOnAChip* 20:3897-4106, 2014.

[43] J.A. White, A.M. Streets. Controller for microfluidic large-scale integration. *ScienceDirect* 3:135-145, 2018.

[44] F. Piraino, F. Volpetti, C. Watson, S.J. Maerkl. A Digital–Analog Microfluidic Platform for Patient-Centric Multiplexed Biomarker Diagnostics of Ultralow Volume Samples. *ACSNano* 1:1699-1710, 2016.

[45] K. Brower, R. R. Puccinelli, C. J. Markin, T. C. Shimko, S. A. Longwell, B. Cruz, R. Gomez-Sjoberg, Polly M. Fordyce. An open-source, programmable pneumatic setup for operation and automated control of single- and multi-layer microfluidic devices. *ScienceDirect* 3:117-134, 2017.