PHY 651: Final Project
Raspberry Pi Pico Accelerometer

Ananya Bandopadhyay [a,1] and Sierra Thomas [a,2]

[a] Department of Physics, Syracuse University, Syracuse, NY 13244, USA.

1 Introduction

The goal for this project is to build a reliable accelerometer, which measures the acceleration of a ping-pong ball as it travels through an evacuated vacuum tube, sealed at both ends. At the initial time, one of the sealed ends of the tube is popped, allowing the air to rush in and launch the ball out. The ball will accelerate due to the force from the air being applied throughout the ball's movement in the tube. Hence, the ball travels with non-constant acceleration, which is imparted due to the impulse from the air rushing in.

To monitor the path of the ball, we use light-emitting diodes (LEDs) as the source of light within the tube, opposed with a set of light-dependent-resistors (also called photoresistors) to detect the change in the amount of incident light as the ball passes between the LED-photoresistor pair. In this paper, we show how we connect this equipment to a pair of Raspberry Pi Picos, which we program to obtain the time that the ball passes through and compare that to the distance between photoresistors. Using the finite-difference method, we can use this obtained information to calculate three acceleration values.

2 System Specifications

In this section, we define the set of conditions that the accelerometer is designed to work in. The system should be able to collect data and calculate values of velocity and acceleration in varying light conditions. Since Raspberry Pis require a specific downloaded IDE, it would be more user-friendly to have them run independently of a computer, which is possible by using a battery pack to power the Raspberry Pi Picos. Ideally, it should stream the output to an LCD screen, so that we can read acceleration values without a computer attached. There should be an automatic boot-up sequence that tests the proper functioning of the LEDs and photoresistors. After the testing sequence, the program will wait to be triggered by the ping-pong ball passing over the first photoresistor.

[1] abandopa@syr.edu [2] sthoma31@syr.edu

3 Instrument Description

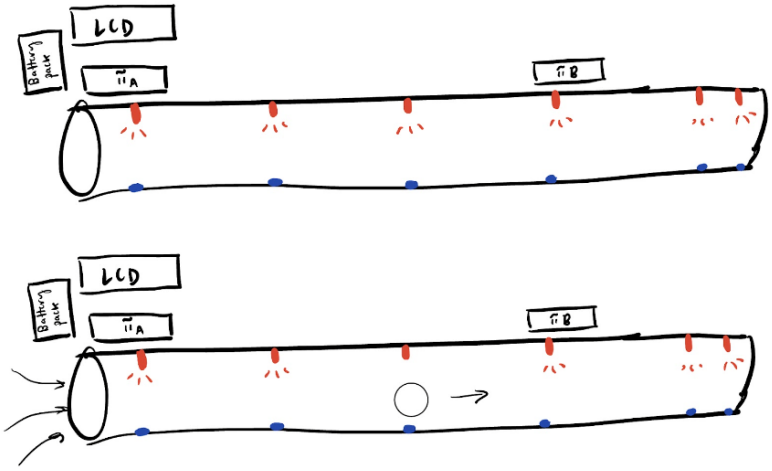

Figure 1: Diagrammatic representation of the model for the accelerometer showing the vacuum tube with the raspberry pi setup, and the ping-pong ball traveling through the tube.

Fig. (1) depicts our schematic for the model. The top diagram represents the vaccuum tube without the ping-pong ball, and the bottom shows the air flowing in and propelling the ping-pong ball through. The LEDs are glowing and placed on top, and the photoresistors have lowest resistance when illuminated. This setup will record the times when the photoresistors individually experience an increase in resistance (when the ping-pong ball blocks the light from the LED).

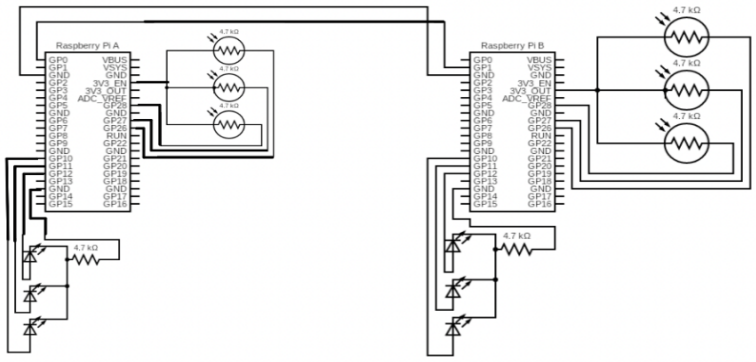

Figure 2: Schematic diagram for the accelerometer showing the different components of the circuit.

Fig. (2) depicts the circuit diagram for the accelerometer. The two Raspberry Pi Picos act as the data acquisition system. The accompanying software we have written uses this data to calculate the velocity and acceleration of the ball at different instances in its trajectory, and displays the output on a screen.

We initially attempted to use photodiodes and transistors to align this project more with the content of the class, but we struggled to get logical true or false voltage values that the Raspberry Pi's GPIO (analog input/output) pins would accept with these. As a workaround, we swapped to photoresistors and the Raspberry Pi's ADC pins.

4 Software and Calculation Methods

Each Raspberry Pi Pico has three analog-to-digital-converter (ADC) pins. ADC pins are capable of converting analog signals to digital signals - our application of this is determining what percentage of voltage was lost while passing over the photoresistor. The Raspberry Pi Pico ADC pins have a 12-bit resolution (although MicroPython's ADC library scales this up to 16-bits). We divided the number received by the total number of bits to get our percentages.

We connect a photoresistor to each of these pins, such that when 4 percent (or more) of voltage is lost, the resistance has gone up from the ping-pong ball blocking the light. One of these pins was used for calibration, to record the amount of light hitting the photoresistors, due to both the LEDs and the ambient light. The time stamps for the ball passing through the remaining data points are recorded and used to calculate the velocity and acceleration of the ball using the finite difference method, i.e.

$$v = \frac{x_{i+1} - x_i}{t_{i+1} - t_i}.$$

$$a = \frac{v_{i+1} - v_i}{t_{i+1} - t_{i-1}}.$$

The location of the photoresistors is specified in advance, and the software uses the data points, i.e. recorded values of time-stamps when the ball crosses the photoresistors, to calculate the value of the ball at different instants. The time is recorded using an in-built MicroPython library that can calculate time in milliseconds from the boot time. Since we used two photoresistors, to maximize the number of ADC pins that could be put to use, we had to set up a communication channel between them, which allowed for the exchange of data between the two. We were able to set this up using a simple python script [1] that can be used for sending messages over UART connections from one device to another.

Our software is written such that the secondary Raspberry Pi (RPP B) captures the calibration light value - as a percent - as well as the milliseconds since the RPP B booted up. RPP B sends the calibration light percent to the main Raspberry Pi (RPP A). Upon receiving this message, RPP A records the milliseconds since it booted up. These time values will be subtracted from the data points taken from both machines respectively, in order to sync the data. We expect a minimal amount of error here, as the information should be sent quickly through the wires.

From there, RPP A waits to receive the first data point. RPP B waits to receive its first data point as well (the fourth consecutive photoresistor in the ball's path). Once both machines have collected their data, they subtract their synced time value. RPP B sends the two data points to RPP A, which performs the calculation above. We attempted to connect RPP A to an LCD screen in order to output the results, but we received a number of obscure errors. The most-likely culprit of these errors is either a), the LCD screen is missing resistors and not functional or b), that the memory space the LCD screen needs is already being used by our interface with RPP B. If b) is the reason, a workaround could be that the Raspberry Pi Picos we used are the W model, which also connect to the internet. We could transmit the results over the internet instead of the screen in the future.

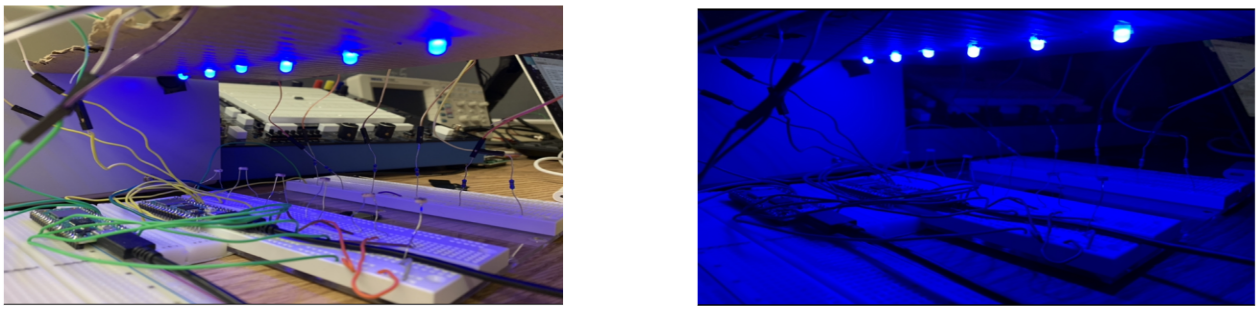

Figure 3: Images showing the accelerometer setup under varying light conditions

5 Results and Next Steps

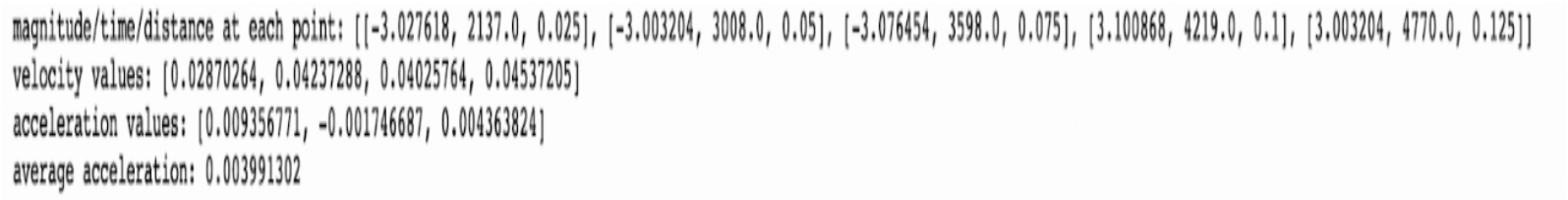

Figure 4: Image showing the accelerometer's displayed output, including velocity, acceleration and average acceleration.

Fig. (4) is a screenshot showing the results from the accelerometer (in SI units) as we tested it using a finger to cross over each photoresistor. The distance between the photoresistors being of the order of a few centimeters, and the crossing time across them being of the order of seconds, the calculated values of velocities and acceleration in ms^{-1} and ms^{-2} respectively can be seen to be fairly accurate. As a note, for this trial run, we removed the LEDs and used a single bright LED as a uni-directional source to avoid interference from nearby LEDs. In the future, we do not anticipate this to be an issue as the vaccuum tube is around 3 meters long and the LEDs will be further away.

Although this project strayed slightly from the content of this class, we did use skills obtained in this class to work with data acquisition tools (the RPPs), photodiodes and transistors during the first attempt, and logic values and memory. Also, we were able to troubleshoot more effectively thanks to this class.

Going forward, we hope to install this setup on the vacuum evacuated tube with the ping-pong ball launched into it, to track its position at various instances in time and use a plotting module to make real-time plots of velocity and acceleration of the ball as it traverses the tube. We also would like to do a quick testing sequence at bootup to blink each LED and be sure the photoresistors are working and aligned - this should not be difficult to do but we were unable to due to time constraints. To make the system independent of a computer, we would need battery packs for the Raspberry Pis. We would also rename the data-collection files to 'main.py' and upon bootup, the Raspberry Pis would calibrate and wait - data collection is triggered when ping-pong passes through the first photoresistor.

The code for programming the RPPs and building this accelerometer is available at:
https : //github.com/sierracthomas/raspberry_pi_project.

References

[1] Kevin McAleer, https : //github.com/kevinmcaleer/easy_comms, easy_comms raspberry pi communication project.

[2] Raspberry Pi Pico Getting Started Tutorial with MicroPython