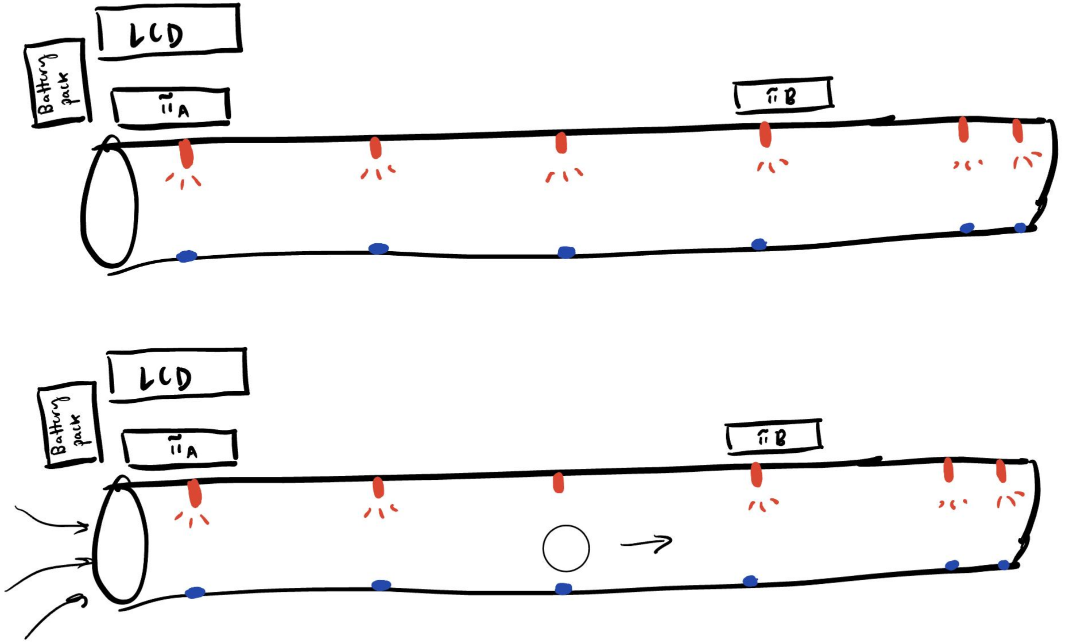
Modern Instrumentation: Raspberry Pi Pico Accelerometer

Ananya Bandopadhyay, Sierra Thomas

Goals:

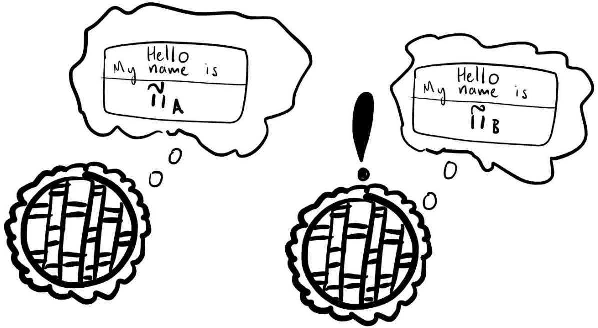
- Build an accelerometer to measure the acceleration of a ping-pong ball as it leaves a vacuum
    - Collect data points when ping-pong ball blocks light sensor
    - Write program to collect data points and calculate acceleration
- Specifications:
    - Needs to work in varying light conditions
    - No dependence on a computer (battery pack and external clock are required)
        - Ideally an LCD screen to show data
        - Automatic boot-up sequence and then wait for data collection to be triggered

Big-picture schematic:



For wiring, see project on github:
https://github.com/sierracthomas/raspberry_pi_project

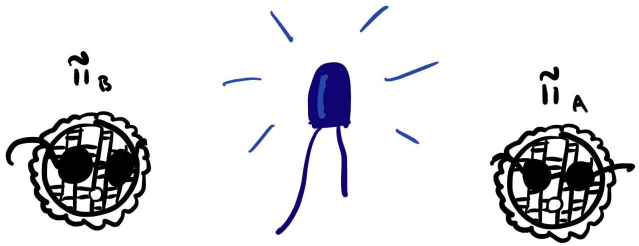
Process - Learning about Raspberry Pi Pico (RPP)



- Raspberry Pi Pico
  - board with a microcontroller chip
  - can handle Python
  - Very cheap
- Learn how microcontroller boards work
- Limitations:
  - MicroPython
    - Will have to manually calculate acceleration
  - Only 3 ADC pins per Pico
    - We had trouble getting 1 or 0s directly from photodiodes or photoresistors
    - One needs to be allocated to a calibration (to account for varying light)
      - Can’t get a useful acceleration with only two points
  - Need to have two picos that communicate with each other

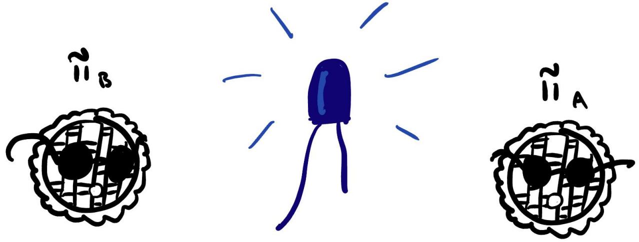
Process - Taking light data with photoresistors

- Tried photodiodes
  - Had issues getting “true” or “false” values
  - Took shortcut by using ADC pins and photoresistors
- Swapped to photoresistors (light-dependent resistor)
  - High resistance when dark, low resistance when light
- With ADC pins we can calculate percent of voltage (received voltage/input voltage)
  - Close to 100% with direct light
  - Drops ~ 4% when we barely block light source so fairly accurate - this is a datapoint



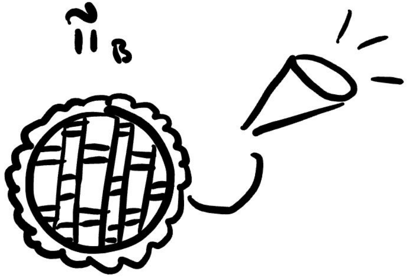
Process - Taking light data with photoresistors, cot'd

- RPP B allocates one pin to "calibrate" light data for varying light conditions
  - When object passes in front of the photoresistor, we calculate light drop relative to the calibration value
  - Leaves 5 usable data points
- Software receives these data points consecutively



Process - Have RPPs communicate with each other

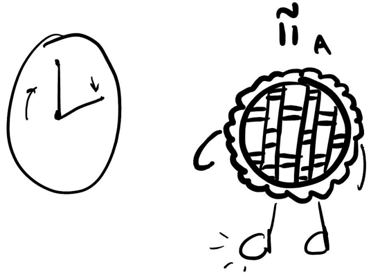
- Found helpful code[1] to send messages from RPP B to RPP A.
- Now we can record data from both
  - Can have RPP A receive and calculate all data

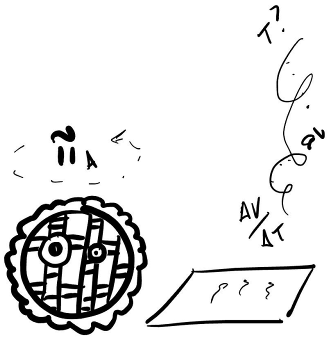


[1]https://github.com/kevinmcaleer/easy_comms

Process - Take data and calculate acceleration

- Brainstormed ways to get time for each datapoint
  - Decided on counting microseconds since booting up, a MicroPython function
- Once RPP A receives the data and stores it into an array
  - Calculate velocity between distance/time points (we get four values)
  - Calculate acceleration between velocity points (get three values)
  - Calculate average acceleration but also display three acceleration points
    - We may want to observe non-constant acceleration





Sum up of project: **hardware**

- Six photoresistors aligned with six LEDs
  - One is for light calibration value, rest are for data
- Three photoresistors and three LEDs connected to each RPP
- LCD screen (not yet) connected to RPP A
- RPP A and RPP B connected and in communication with each other

PICTURE OF SETUP

Sum up of project: **software**[2]

- RPP B sends light calibration value
    - Both RPPs record when this value is sent or received (microseconds since bootup) to sync machines
- RPP A initializes data array
    - Distance values are preset based on configuration
- RPP A collects three data points (time in milliseconds) when more than 3% of light blocks it
- RPP B collects two data points and sends to RPP A
- RPP A calculates acceleration from data array and converts to m/s^2

SHOW PICTURE OF OUTPUT ARRAY?

[2]https://github.com/sierracthomas/raspberry_pi_project

Applications to this class

- Learning how to use DAQ
  - Analogous to the RPP taking data
- Photodiode/transistor (during attempt) and diodes
- Some of learning about memory, storage, boolean values from analog signals
- Problem-solving!

Results

- Able to get acceleration values!

PICTURES

Next Steps

- Quick bootup sequence to be sure photoresistors and LEDs work, are connected, and can take data
  - Ran out of time to set this up
- LCD screen:
  - Had issue connecting the RPP A to an LCD screen
  - Errors are obscure and not much help
    - Think we're using that memory location for something else, like having the RPPs communicate
- Battery pack for RPPs
  - If program is called `main.py` then will start automatically at bootup
    - Data collection is triggered when ping-pong passes through first photoresistor
- Actually connect to ping-pong launcher