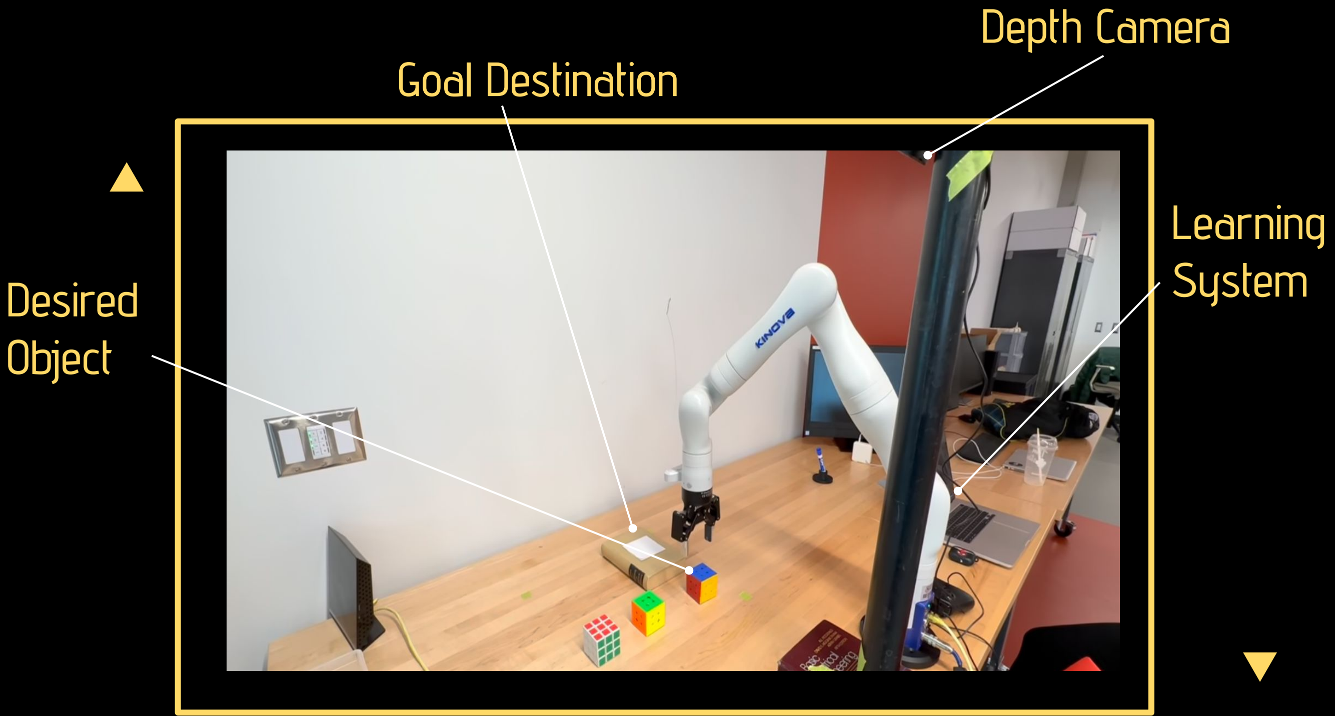
01

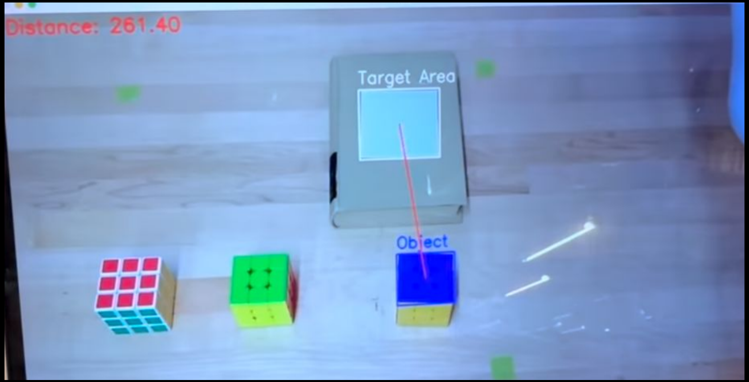
Robotic pick & place system

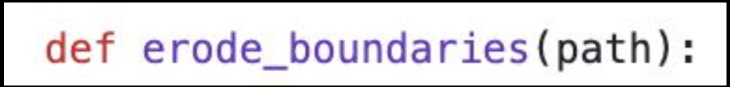
Developed in a team of four

PLACEMENT

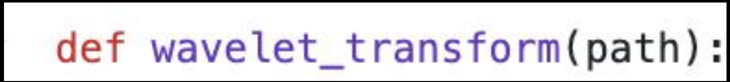

Goal Destination
Depth Camera
Learning System
Desired Object
KINOVA

LIVE FEED - OPENCV

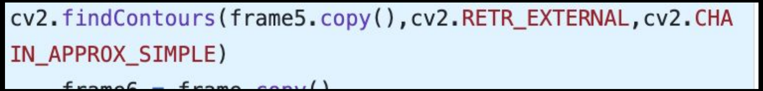




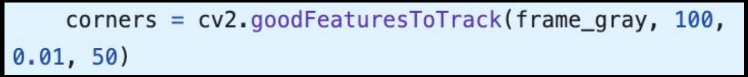
```
def erode_boundaries(path):
```



```
def wavelet_transform(path):
```



```
cv2.findContours(frame5.copy(),cv2.RETR_EXTERNAL,cv2.CHA
IN_APPROX_SIMPLE)
```



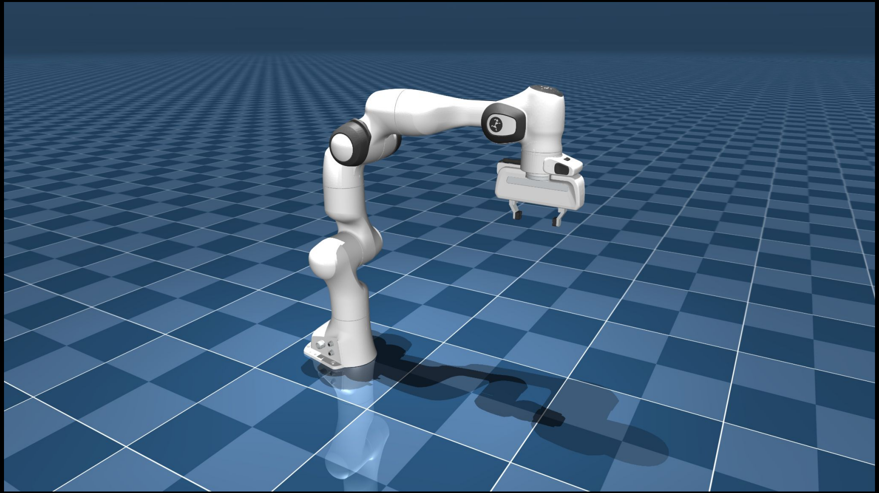
```
    corners = cv2.goodFeaturesToTrack(frame_gray, 100,
0.01, 50)
```

Logical

- Target area is set in advance
- Moving objects are detected
- Distance is found from target area to object
- Depth is found from the hand of the arm and the object

Math

- Utilized openCV to call on linear algebra functions on the array version of the image feed
- Methods included: filtering the array, finding contours and features to track in order to find distance



ROBOTIC TRAINING

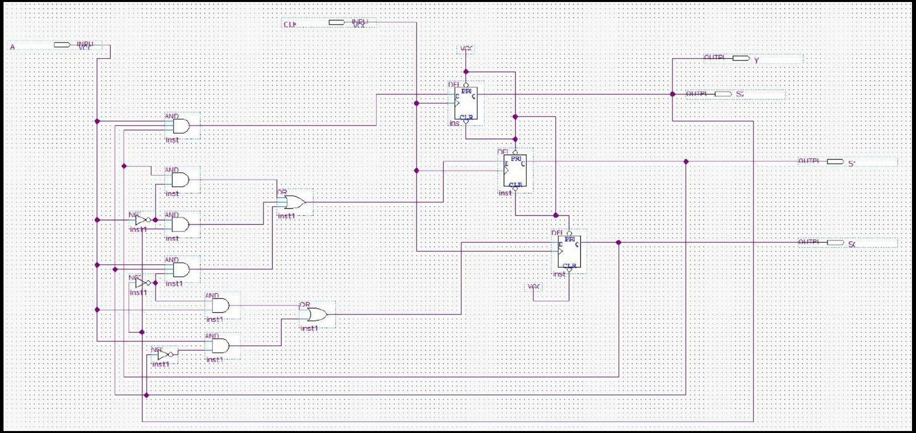
We used MuJoCo Franka Emika Panada to simulate the robot based off of the regression learning the other team did. To feed the decision made by the RL model was done through the XML

02

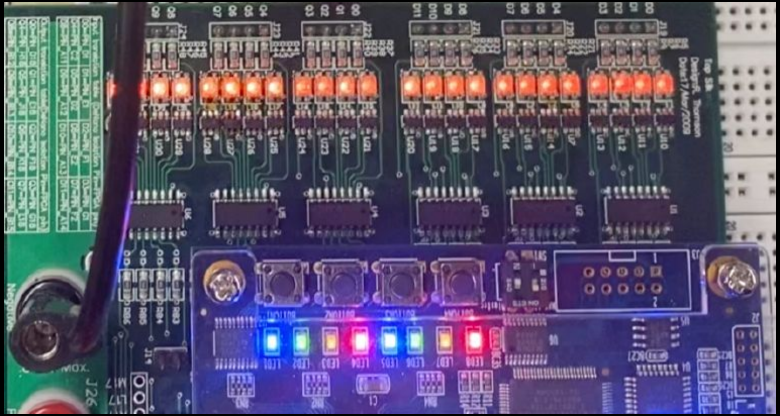
RISC-V and Hardware Organization

Taught in my computer organization and Digital Circuits class

Synchronous Detector Circuit

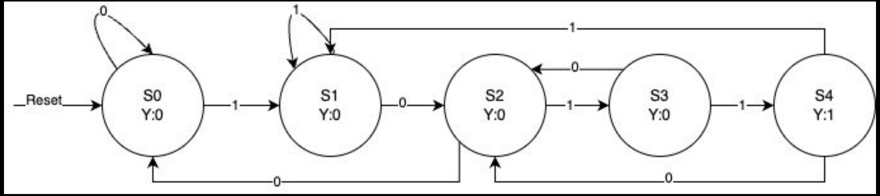


Quartus Schematic



MKS Board Setup

In the completed project, the state bit values (S2S1S0) and the output value Y have been recorded at specific times and entered into a timing diagram. The sequence consistently initiates from state 000, automatically resetting when the programmer is executed. Manual resetting is also feasible using the CLK and A inputs if necessary.



State Transition Diagram

ASSEMBLY ASSIGNMENT

https://github.com/nessmamd/RISC-V

Attached in the GitHub, I have two examples:

I have a strong affinity for coding in assembly, particularly due to the intriguing bit operations that occur even in simple statements like if or jump.

1

Creating bitwise shift, bitwise, and logical operators

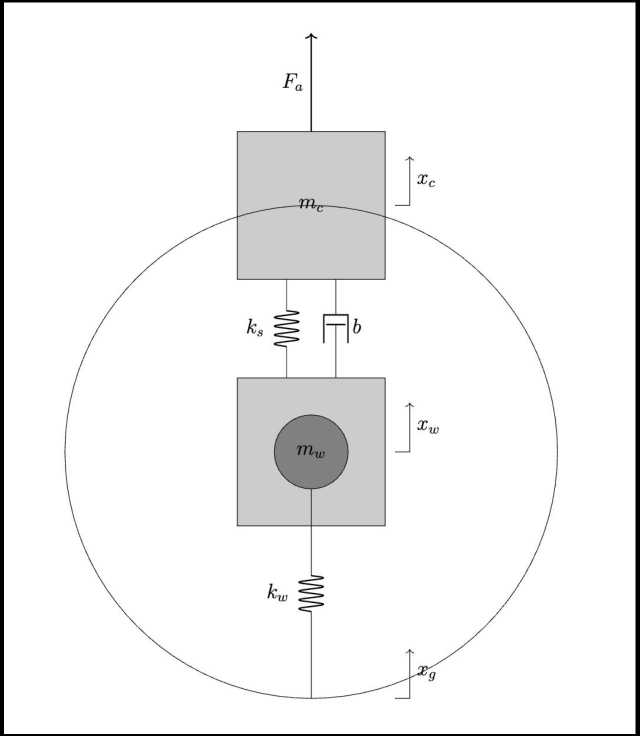
2

Creating lbu and sb instructions

03

PID Controller

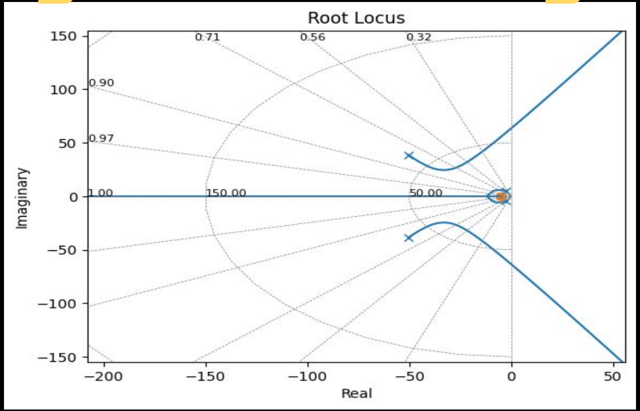
An implementation for the wheel on a space rover. My role was to design a PID controller to attempt to reduce the effect of disturbances (bumps in the road) on the car and my partners was to implement it on a circuit.



Free body diagram

$$\begin{bmatrix} \frac{d}{dt}\Delta x_w(t) \\ \frac{d^2}{dt^2}\Delta x_w(t) \\ \frac{d}{dt}\Delta x_c(t) \\ \frac{d^2}{dt^2}\Delta x_c(t) \end{bmatrix} = \begin{bmatrix} 0, & 1, & 0, & 0 \\ \frac{k_s}{m_w} - \frac{k_w}{m_w}, & \frac{-b}{m_w}, & \frac{k_s}{m_w}, & \frac{b}{m_w} \\ 0, & 0, & 0, & 1 \\ \frac{k_s}{m_c}, & \frac{b}{m_c}, & -\frac{k_s}{m_c}, & -\frac{b}{m_c} \end{bmatrix} \begin{bmatrix} \Delta x_w(t) \\ \frac{d}{dt}\Delta x_w(t) \\ \Delta x_c(t) \\ \frac{d}{dt}\Delta x_c(t) \end{bmatrix} + \begin{bmatrix} 0, & 0 \\ \frac{k_w}{m_w}, & 0 \\ 0, & 0 \\ 0, & \frac{1}{m_c} \end{bmatrix} \begin{bmatrix} F_a(t) \\ x_g(t) \end{bmatrix}$$

Dynamic Analysis

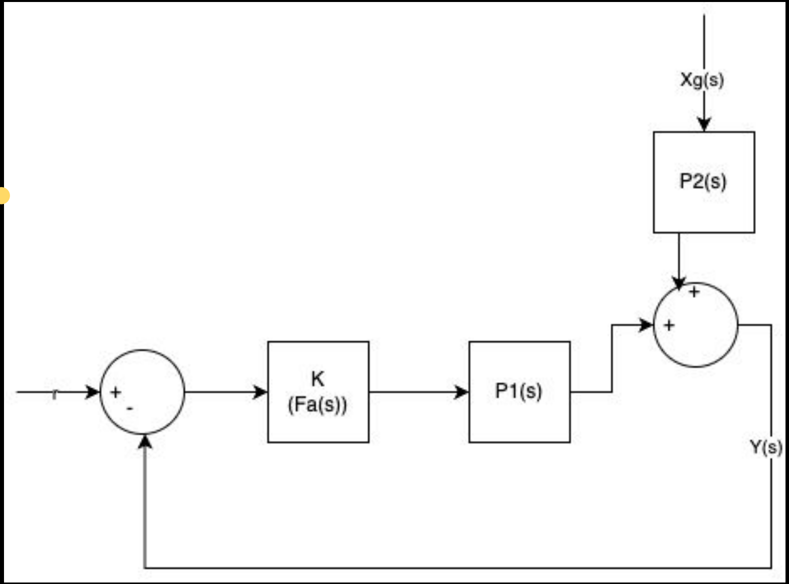


Root Locus Plot

LOOP SHAPING

Based on

Initially, we analyze our feedback loop, converting the state space to transfer function representation, where Fa serves as the controlled input and xg as a disturbance. We then depict the closed-loop system diagram labeling P1 and P2, enabling us to express E(s) = R(s) - Y(s) in terms of R(s) and Xg(s).



Current development.... creating circuit

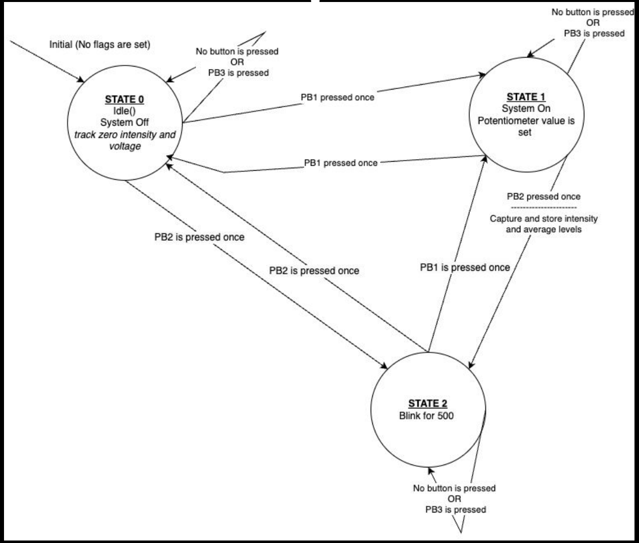
04

Light Controller

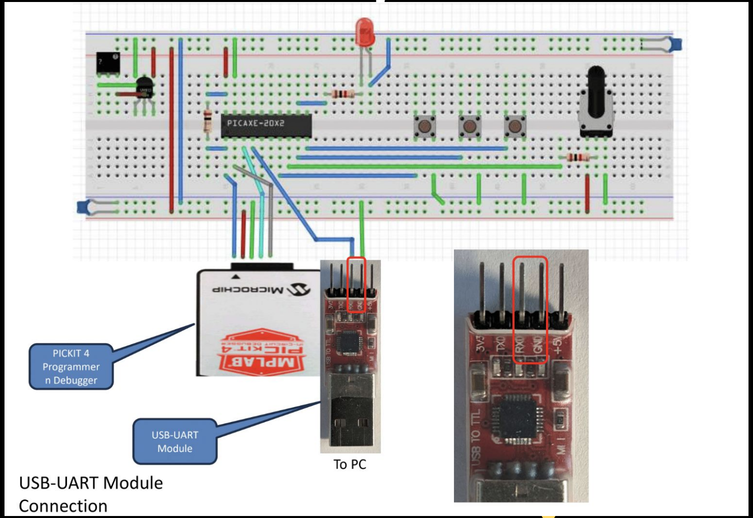
A LED light control app is designed using a microcontroller, UART, and Timer peripherals. It utilizes push buttons (PBs) to toggle between ON/OFF modes and blinking modes for the LED connected to pin 12, while capturing and plotting intensity levels and PWM voltage output vs. time using Python for analysis and visualization.

Prep

State Transition


Initial (No flags are set)
No button is pressed OR PB3 is pressed
STATE 0
Idle()
System Off
track zero intensity and voltage
PB1 pressed once
No button is pressed OR PB3 is pressed
STATE 1
System On
Potentiometer value is set
PB1 pressed once
PB2 pressed once
Capture and store intensity and average levels
PB2 is pressed once
PB2 is pressed once
PB1 is pressed once
STATE 2
Blink for 500
No button is pressed OR PB3 is pressed

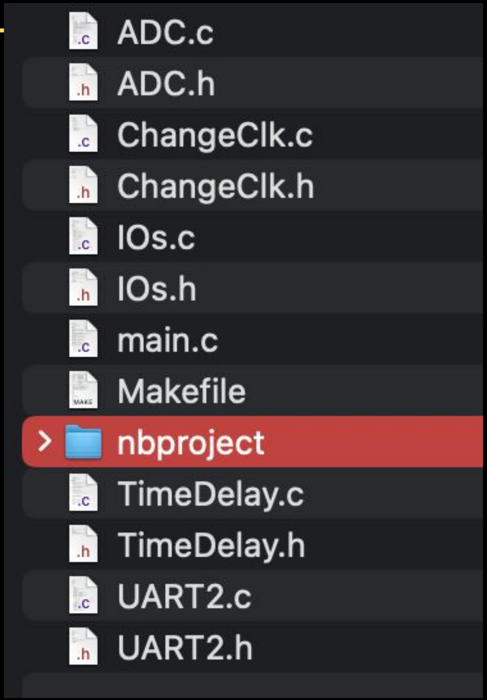
Schematic Diagram



What was done?

Coded on MPLAB X IDE

- ***ADC*** ->This code initializes and performs analog-to-digital conversion (ADC) on pin 8/RA3/AN5 using the ADC module, returning a 10-bit ADC value in unsigned integer form.
- ***ChangeClk*** -> This code snippet defines a function NewClk that changes the system clock frequency based on the input parameter clkval
- ***UART*** -> This code initializes and manages UART2 communication on PIC facilitating transmission of characters, hexadecimal and decimal numbers, as well as strings over UART to an external device.
- ***IOs*** -> This code handles button presses (PB1 and PB2) to control system states using interrupts, enabling functionalities like clock switching, UART communication, ADC sampling, and LED blinking.
- ***TimeDelay*** -> This code implements time delay functions in milliseconds and microseconds using Timer 1 and Timer 2 interrupts, configuring the timers and interrupts accordingly for precise timing.


ADC.c
ADC.h
ChangeClk.c
ChangeClk.h
IOs.c
IOs.h
main.c
Makefile
nbproject
TimeDelay.c
TimeDelay.h
UART2.c
UART2.h

05

Long time No Crypto

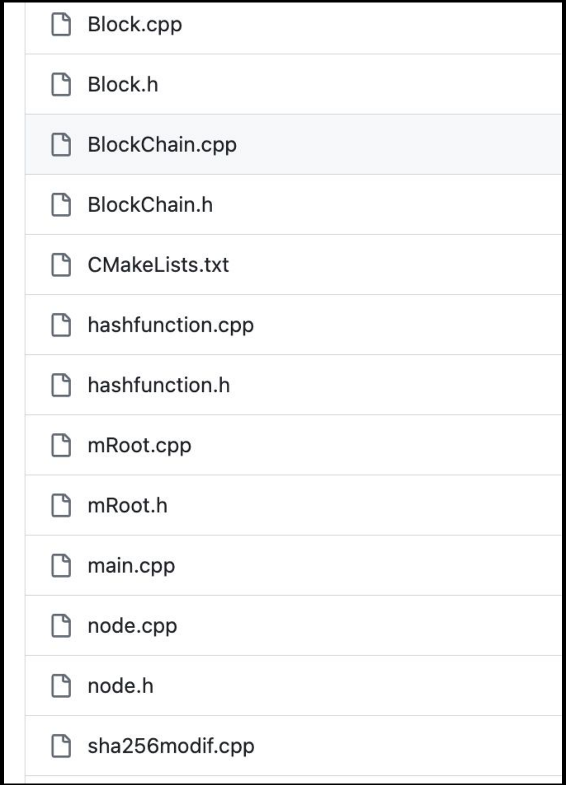
NEWS

Demo: https://www.youtube.com/watch?v=fZaP-H-yOZc
Repo: https://github.com/nessmamd/longTimeNoCrypto?tab=readme-ov-file

Overview

Long Time No... is a unique cryptocurrency generated from messages users wish they had sent to others, which can only be received from someone who desires to share a secret message. The operation involves hash creation through Hill cipher encryption, utilizing SHA256 with slight modifications. Additionally, files for Merkle tree creation, proof, and blockchain entity, alongside socket operation files for client and server-side functionalities, are provided.

Code Division


Block.cpp
Block.h
BlockChain.cpp
BlockChain.h
CMakeLists.txt
hashfunction.cpp
hashfunction.h
mRoot.cpp
mRoot.h
main.cpp
node.cpp
node.h
sha256modif.cpp

Code Hierarchy

- Every block was created as an instance based on the message.
- Blockchain is the connection of the blocks
- Connections are made from nodes
- mRoot, hash function, and sha256 create the encryption (designed by myself)