FPGA Robot Arm Assistant

Jack Erdozain, Nicholas Klugman, Mark Vrablic
Fall 2017 6.111 Final Project Proposal

1 Introduction

1.1 Objective Overview

- Build a robotic arm with camera and projector at the end. Project keystone-corrected incoming notifications from a connected computer. Create a puck with multiple lights that will control the arm and therefore location of the projection. Have the projector follow the puck: this involves CV to locate the puck and controls for the robot arm for it to move as the puck moves. Scroll notification text by rotating the puck using computer vision.

1.2 Goals

Commitment:

CV:

- Pipe video feed into frame buffer
- Identify X, Y coordinates of 3 LEDs on puck
- Generate “center-of-mass” X, Y of LED’s

Robotics:

- Use CV output to generate pose of arm/projector to track puck in one-dimension
- Command servos to move arm to match this pose

Keystone correction/software:

- Display vector graphics based notification icons
- Correct keystone based on the angle of the robotic arm
- Receive notification pings from a connected computer

Goal:

CV:

- Convert X, Y coordinates into a X, Y ,Theta to be used as commands in different modules.

Robotics:

- Move projector to track puck in two-dimensions on surface
- Minimize “jiggle” of arm by placing minimum error bounds for movement

Keystone correction/software:

- Receive and display full text of notifications from computer
- Scroll through notification text using the puck's theta from CV

Stretch:

CV:

- Identify when LED's are covered as a means of additional input (clicks).

Robotics:

- Adjust arm dynamically to maintain constant projection size, or adjust projection size based on commands from the FPGA button inputs
- Implement rudimentary PID controller for control of angles based on error from CV.

Keystone correction/software:

- Add Sound to give personality
- Transmit images from computer to display when no notifications are available

2. Design

2.1 Physical Components

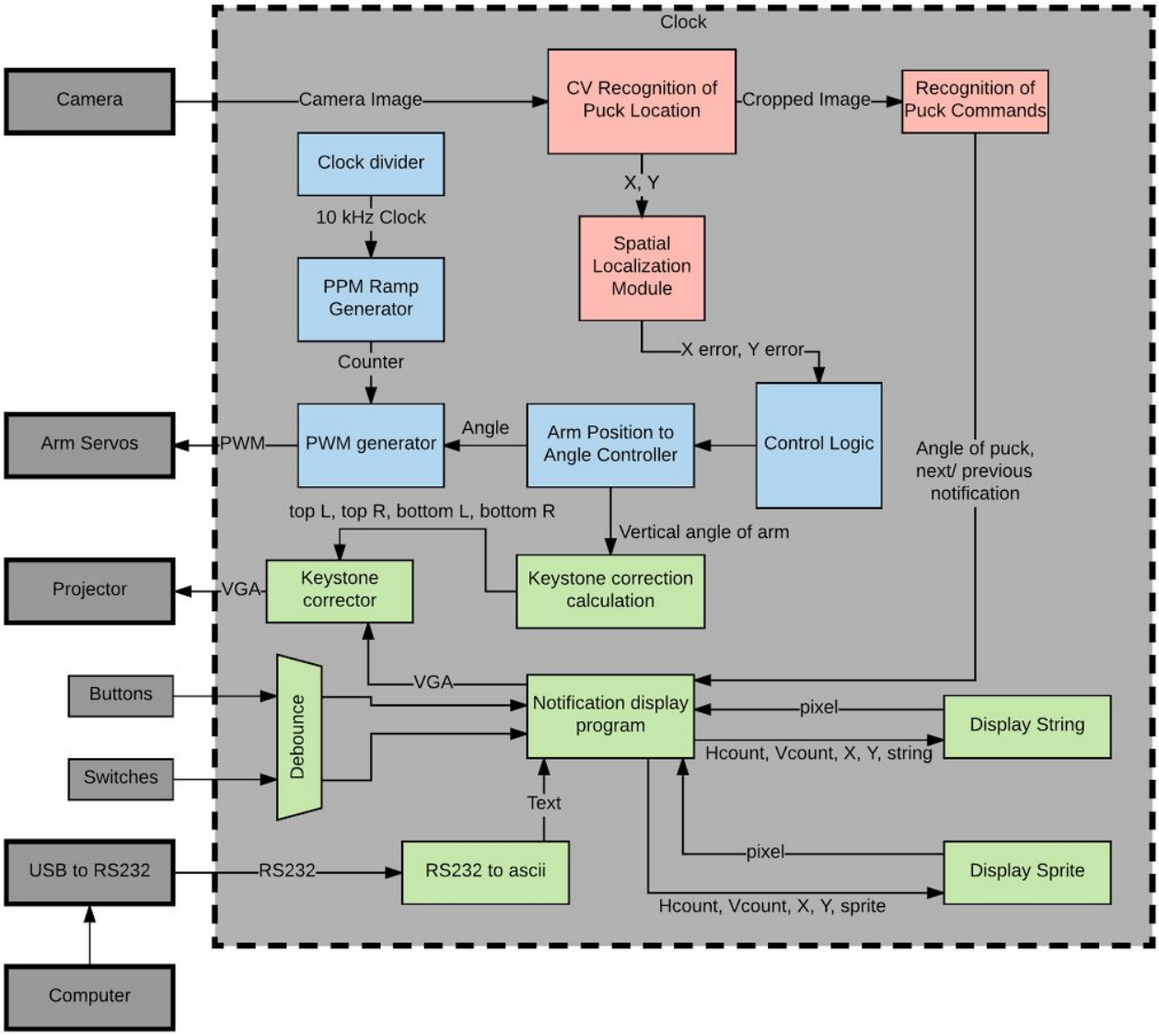
- **Robotic Arm** - An off the shelf robotic arm capable of holding our pico projector on the end and external control.
- **Pico Projector** - A small laser projector to display the program on
- **Cursor/Puck** - An object with IR LEDs which can be moved while still easily viewed by the camera
- **Camera** - RGB camera for CV capable of being attached to the robot arm

2.2 Design Challenges

- **CV** - Identifying the puck markings instead of other environmental factors may be difficult.
- **Keystoning** - Assumes that the servos are exactly where commanded, so error in their position will affect keystoning.
- **Mechanical** - Making a mechanically sturdy system is always a factor to consider in these sort of projects.

3. Implementation

3.1 Block Diagram



3.2 Modules

- **CV Recognition of Puck Location:** Looks for a specific predefined marker that identifies the puck controller. It outputs an x/y “center of mass” on the camera input. We will test this by displaying the camera data with the detected puck location indicated by a superimposed puck.

- **Spatial Localization:** Identifying the puck location on the camera video output is not enough alone. The system must be able to process the changing video input to determine the change spatial location of the puck on the desk and thereby an error in its own positioning so that it may adjust its position and the resulting projector output. We can test this by displaying the error on the FPGA numerical display with the robot fixed; No error should be detected when we place the puck in the correct position, but when we move the puck, it should produce error values corresponding to our movement of the puck.

- **CV Recognition of Puck Commands:** Looks for a specific predefined markers and identifies cursor commands. For example if a marker is covered this will indicate a “click” on the mouse, or if the cursor is rotated it will indicate a “scroll” command. We can test this by displaying on the numerical display a code saying which commands are active (i.e. bit 0 high when left-click active and low when inactive).

- **Clock Divider:** Slows down the clock for the PWM Ramp Generator. Can be tested with oscilloscope.

- **PWM Ramp Generator:** A sawtooth generator must be used to create a global pwm clock and help in generating pwm. We can test this by using a logic analyzer on the output bus to verify that the ramp is incrementing properly.

- **PWM Generator:** Takes in the Ramp input and position commands, and generates a duty cycle based on a comparison of ramp value and commanded angle. We can test this using the oscilloscope.

- **Arm Position to Angle Controller:** Takes in desired XYZ position and with an internal knowledge of arm geometry generates desired angles of servos. We can display the value being fed to the PWM generator on the Nexys 4 display. The value is also fed into the keystone module

- **Control Logic**: Uses a combination of input on the desired behavior (holding position, etc) and the spacial data from the camera to determine where the robot should move and how the mouse should behave. We can test this by feeding in a dummy location and outputting the x, y and z

positions on the Nexys 4 display and flash the built in LEDs with the cursor commands being output.

- **Keystone Corrector:** Takes in VGA and outputs a new, keystone corrected VGA signal. It calculates the pixel locations based on the difference in the top and bottom corners from the “Keystone Correction Calculation” module. This can be tested by using any example VGA program as input and the VGA monitor as output.

- **Keystone Correction Calculation:** for the corners of the keystoned image based off of the angle information from the “Arm Position to Angle Controller” module. This can be tested by feeding in known static angles and displaying the output points over VGA

- **Notification Display Program:** Takes ASCII characters from a connected computer and displays them along with any associated graphics. It then outputs the image over VGA, where it is used as an input to the keystone corrector module. It can be tested using a normal VGA monitor.

- **RS232 to ASCII:** Takes an incoming serial connection from the computer and outputs received text in ASCII character codes. This can be tested by showing incoming data on the Nexys 4’s built-in display.

- **Display String:** Takes in ASCII character codes in a string and provides a pixel output when the corresponding text is at (hcount, vcount).

- **Display Sprite:** Takes in a position for the sprite and provides a pixel output when the corresponding sprite is at (hcount, vcount). There will be one of these for each sprite we wish to use.

Project Timeline

	CV	Robotics	Display	Class Deadlines
Week of 10/29	Pivot from laser projector to this project			Block diagram conference
Week of 11/5	Get video feed piped to VGA for debugging	Get arm hardware working	Keystone sprites	Project Design Presentation
Week of 11/12	Recognized IR clusters and Assign X, Y positions	Generate PWM from commanded angles	Add text + create vector icons	Checkoff Checklist, Proposal Revision
Week of 11/19		Calculate proper angles for a given change X error and Y error	Read serial from computer for notification, debug core program if necessary	
Week of 11/26	Angle of Puck	Merge parts together + debugging		
Week of 12/3	Debugging			
Week of 12/10				Project checkoff, demo, and report