Project FLOW: Fun Low-Power Observer-Interactive Waterfall

Ben King, Connor Heckman, Jack Gray, Robert Perkins
College of Electrical and Computer Engineering
University of Central Florida
Orlando, FL, USA

***Abstract* — This paper presents the design of a scaled-down user-interactive waterfall feature, intended to be implemented as a peripheral to a full-scale solar sculpture. This waterfall is made up of a graphical array of water solenoid valves and a series of LED strips which can be manipulated through observer interaction. Both audio input through the sculpture's sound card and the physical observer motion detected by the installation's optical sensor, can be used to produce attractive and entertaining patterns in the installation's solenoid array.**

***Index Terms* -- Interactive systems, Machine vision, MOSFET circuits, Object segmentation, Solenoids**

I. INTRODUCTION

Orlando Utilities Commission in partnership with the Tavistock Company tasked our group with designing a solar sculpture peripheral that would attract attention and generate excitement about the structure while educating about the practicality of solar power. The proposed solar sculpture will be placed in the center of the town square for high visibility, and in line with the mosaic theme of the surrounding buildings.

Interested in the cross over between art and engineering our group of Jack Gray, Connor Heckman, Ben King, and Robert Perkins began designing a project to highlight both aspects. To symbolize the flow of energy from the sun, harvested by the solar panel and redistributed to power the solar sculpture we decided on an interactive waterfall. The interactive waterfall works by receiving an input signal from the mounted camera, then translates the horizontal position of the person to the corresponding solenoid valves to create a falling stream. The mounted CMUcam5 receives color graded input and sends this information to the Raspberry Pi 3 for processing. The Raspberry Pi system control unit compares the current input image to a reference image to determine the horizontal and vertical coordinates of the person interacting. These coordinates are output through the MSP430 microcontroller to two 8-bit shift registers. These shift registers send enabling signals to the solenoid switching circuit on the printed circuit board.

Powered by the 12V power supply and controlled by the signals send to the switching circuit the solenoids are turned on in relation to the horizontal position of the observer, while the vertical position corresponds to the height of the LED strips illuminated. To further the entertainment value of the water and light display, an audio input sensor is used to make the changing streams of water respond to songs being played. The water pump system uses a single 12V pump to supply a constant pressurized flow of water to the solenoids in a closed loop systems.

The power supply requirements are split into two sections: the grid-tied energy collected from the solar panels, and the energy distribution from the grid into the system without the use of batteries. The photovoltaics convert the solar energy into DC electricity through a DC to DC converter with Max Power Point Tracking. For grid interconnectivity the DC electricity is pumped back into the grid using a DC to AC inverter. The final deliverable for this project is a one eighth scale model of the final implementation of the complete solar sculpture. The size restraint will eliminate the possibility of implementing grid-tied solar panels into the final deliverable of our design. The power supplied to the project is from a grid-tied AC outlet that is then converted into 12V to power the solenoids, LED lights, and water pump. This 12V supply is then passed through a 5V regulator to supply the power used for the Raspberry Pi, CMUcam5, and Audio input sensor. Further voltage regulation down to 3.3V is necessary for powering the MSP430 microcontroller.

As the project design developed the requirements and limitations of the OUC sponsorship changed. Originally the use of an interactive feature using water was approved on the basis that the cost of construction and maintenance will remain low. After the presentation of our prototype waterfall design concerns with maintenance costs caused our project to not be selected for the final implementation of the solar sculpture. We continued designing this entertainment feature for possible use and consulted the art and mechanical team for the design of an approved low maintenance solar sculpture

The water system will consist of a reservoir at the bottom of the structure for holding water, a submersible water pump, water supply and return tubes to provide water to the top of the structure, the solenoid valves that control each individual water stream, and the solenoid switching circuit which drives the solenoid using the signals from the solenoid control unit.

After viewing previous waterfall designs utilizing solenoid valves, it was determined that a spacing of 32 valves per meter should produce an acceptable horizontal resolution for the waterfall display. Due to budget constraints and the cost of solenoid valves, the scale model will use a half meter wide array consisting of 16 solenoid valves.

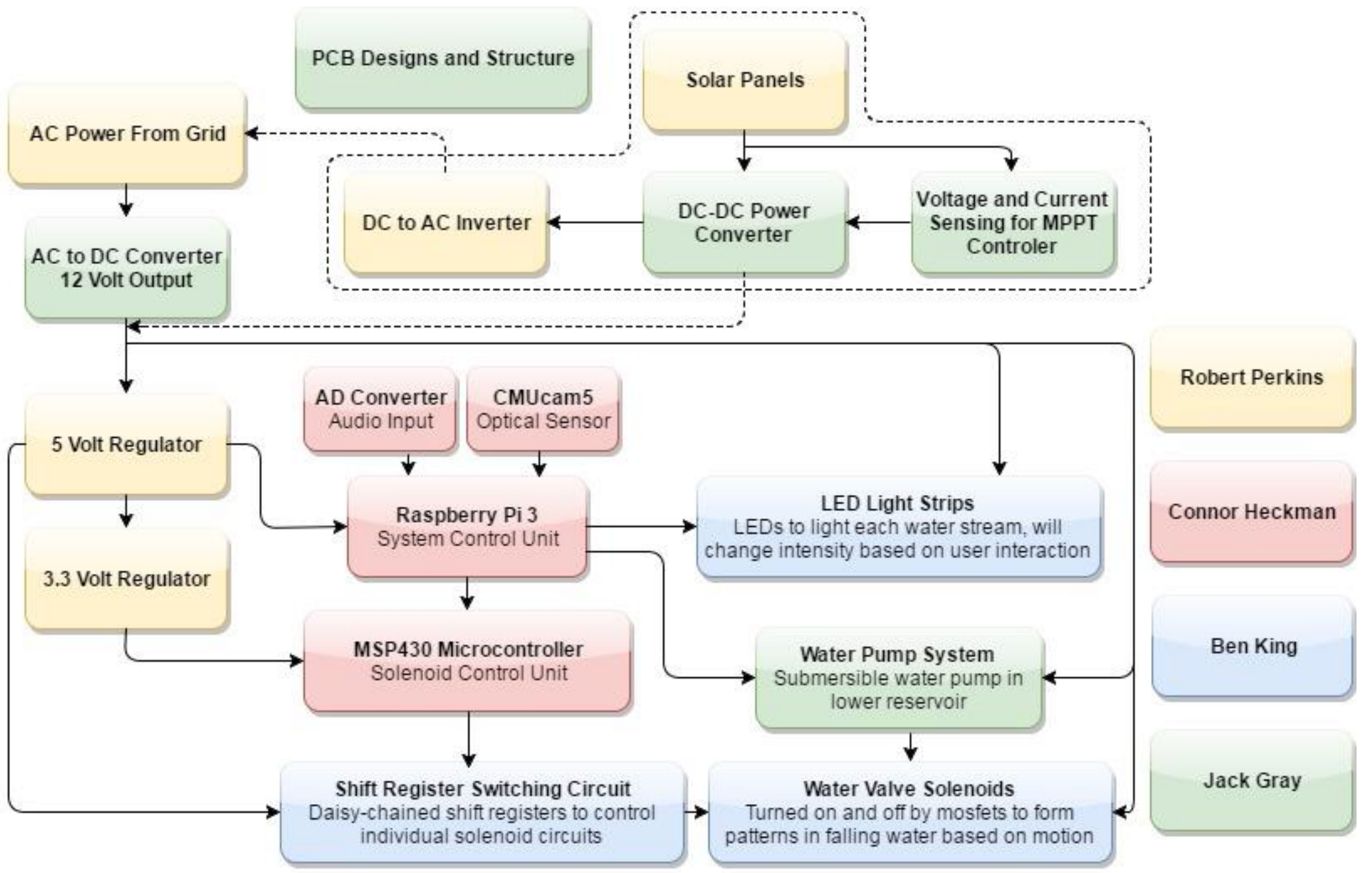
PCB Designs and Structure
Solar Panels
AC Power From Grid
DC to AC Inverter
DC-DC Power Converter
Voltage and Current Sensing for MPPT Controler
AC to DC Converter 12 Volt Output
Robert Perkins
AD Converter Audio Input
CMUcam5 Optical Sensor
5 Volt Regulator
Raspberry Pi 3 System Control Unit
LED Light Strips LEDs to light each water stream, will change intensity based on user interaction
Connor Heckman
3.3 Volt Regulator
MSP430 Microcontroller Solenoid Control Unit
Water Pump System Submersible water pump in lower reservoir
Ben King
Shift Register Switching Circuit Daisy-chained shift registers to control individual solenoid circuits
Water Valve Solenoids Turned on and off by mosfets to form patterns in falling water based on motion
Jack Gray

Fig.1 Hardware Block DiagramWaterfall Feature Design

A. *Solenoid Valve Selection*

In order to precisely control the falling water streams, a solenoid valve is needed. A solenoid valve is a valve that is electromechanically controlled by use of an electric current passing through the solenoid. The solenoid itself is a coil of wire that is wound around a metallic core. When current is passed through the coil, it creates an electric field which moves the core pin. In a solenoid valve, this movement is used to control the opening and closing of the valve.

Many solenoid valves that are approved for use with water are made from plastic or brass, but due to cost constraints, the main focus for this project is on plastic solenoid valves. The solenoid valves have several other specifications to compare, such as response/opening time, normally open (NO) or normally closed (NC) operation, valve diameter, and their operating voltage and current draw. For response time, a fast response is needed for the opening and closing of the valves in order to produce a decent resolution from the falling water. An acceptable response time for this application is less than 100ms, but lower response times would improve performance, as the vertical resolution quality relies on the valve response time.

Given that this project will be run with solar power in mind, the lower 12 volt DC operating voltage solenoids were considered rather than AC or 24 volt DC valves. Based on these specifications, a relatively inexpensive nylon constructed water solenoid valve made by US Solid was chosen for this waterfall display application that included a normally closed (NC) operation, ¼ inch diameter, 12 volt DC operation with a 20ms valve response time. The US Solid brand 12 volt solenoids are rated at 4.8 watts each.

B. *Switching Circuit Design*

A switching circuit is designed in order to control each of the 16 solenoid valves using the MSP430 output pins. The MSP430 does not have enough output pins to control the 16 individual valves by itself, so this issue was solved by using two daisy-chained 8-bit shift registers that are hooked up to the MSP430. The shift register allows serial data from the MSP430 to be converted to 16 parallel data outputs, with a separate 5 volt pin to control each solenoid valve.

The next consideration was how to control the 12 volt solenoids using the 5 volt signals from the shift register pins. Using a transistor as a switch was an obvious solution, such a BJT or a mosfet. While a BJT device would have worked fine for this application, the nature of the base current and base resistor required meant excess power dissipation. While this power dissipation is relatively small, it is still taken into consideration given the solar powered nature of the full-scale design. Due to the nature of the mosfet's insulated gate and low drain to source resistance, the power dissipated by the mosfet is significantly less than a BJT. For these reasons, an n-channel power mosfet was chosen that can handle up to 30 volts, with a low gate threshold of only 1.8 volts, allowing the 12 volt solenoid valves to be controlled by the 5 volt shift register pins.

The final consideration for the circuit design was flyback voltage. Due to the inductive nature of the solenoid valve, the sudden reduction in current when the mosftet is switched off creates a large voltage spike of negative polarity across the solenoid load. The relationship is given by the voltage equation for an inductive load as shown in (1).

$$V = L\frac{di}{dt} \qquad (1)$$

This high voltage spike could arc to a nearby ground trace on the PCB or damage the mosfet device itself. In order to prevent this, a simple 1N4001 diode is placed in parallel with the solenoid valve, positioned so that it blocks current flow through the diode during normal operation, allowing it to flow through the solenoid as normal when the mosfet is on. When the mosfet is switched off, and the solenoid valve produces a negative polarity voltage spike, and the diode is then forward biased, allowing the current to safely dissipate back through the load, rather than arcing or damaging the mosfet device.

Given all of the previous design considerations, a schematic of the final solenoid switching circuit can be seen in Figure 2.

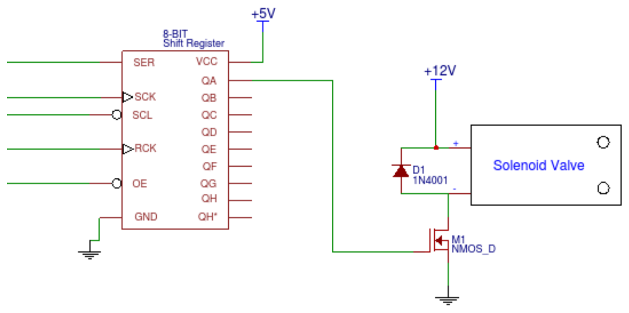

Fig. 2 Solenoid switching circuit schematic

C. *Water Reservoir and Pump*

With the solenoid valve selected and the switching circuit designed to control the valves, a water delivery system was designed next in order to provide constant water flow to the 16 solenoid valves. Consideration had to be given to make sure that the water flow was constant, while providing the same pressure regardless of whether only a single valve was open, or all 16 valves were open.

For this design, a water system was designed consisting of an upper reservoir and a lower reservoir. The solenoid valves are directly connected to the upper reservoir, which supplies constant pressure to each valve. The lower reservoir catches the falling water from the solenoid valves, which is recirculated back to the upper reservoir using a small 12v pump.

A 4 inch PVC pipe was chosen to act as the upper reservoir. Through testing, it was determined that 4 inches of standing water provided enough pressure from gravity to provide a steady water stream from the solenoids. By using pressure from gravity, this eliminated the need for more complex water pressure calculations, and made the requirement for the upper reservoir to simply be completely full at all times in order for constant pressure to be supplied regardless of the number of valves open.

In order to keep the upper reservoir full at a constant water level, two ½ inch flexible water tubes were attached to the top of the upper reservoir tube, with one on each side. One tube is connected to the 12v submersible pump, and acts as the water supply tube. The second tube acts as the water overflow and return tube, which returns excess water pressure to the lower reservoir. A simple diagram of this water supply system can be seen in Figure 3.

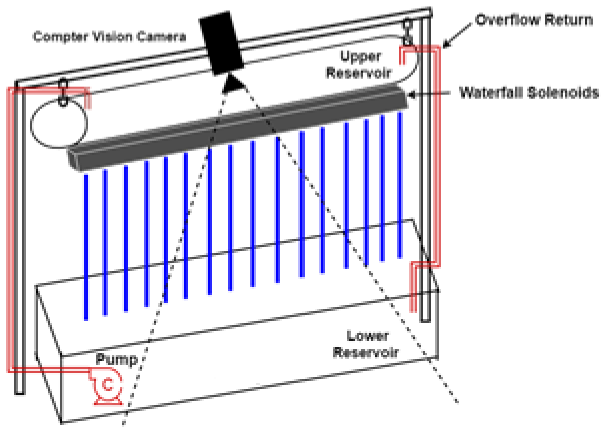

Fig. 3 Interactive waterfall diagram.

The use of both water supply and overflow return tubes allows for all air to be purged from the system, and provides a constant full water level in the upper reservoir. This constant full water level ensures constant pressure to each solenoid valve regardless of how many are open or closed.

II. System Control

The Installation is programmed with three separate "modes" of entertainment. First is the Aesthetics mode, which consists of the installation looping through a series of preprogrammed routines designed to impress observers and draw attention to the structure. Second is motion interaction mode, during which observers of the installation will be able to control solenoid and LED output by moving in view of the installation's optical sensor. Last is Audio Interaction mode, where observers can plug their phones in to the Installation's audio jack and watch as the installation performs music visualization through the use of the solenoid array.

Mode switching is performed through the use of a remote controller (iPazzPort Wireless Mini Keyboard) operated by the system administrator. By hitting designated hotkeys on the remote, the system administrator calls or terminates executables for the distinct modes via a shell script which is continuously running within the System Control Unit (Raspberry Pi 3).

III. Computer Vision

Computer vision plays a fundamental role in the in the installation's motion interaction mode. In this mode, the waterfall's optical sensor captures sequential images of the foreground using a Pixy Camera (cmuCam5) at a rate of 20 frames per second. This frames are smoothed to reduce noise and are then passed to the Raspberry Pi 3 which interprets observer motion and sends output signals to the Solenoid switching array and the variable brightness LED strip.

A. *Pixy Camera*

The Pixy cam, or CMUcam5 is an inexpensive DIY vision sensor used for object recognition purposes. It contains a NXP LPC4330 dual core onboard processor and hence can handle the intensive processing required to recognize objects frame by frame, sending positional data to the microcontroller running the camera.

The Pixy Cam is built to run in concert with an Arduino microcontroller, however over the course of testing, an Arduino Uno microcontroller was found to have insufficient processing power for the installation's needs. A raspberry Pi 3 was used to control the Pixy Camera instead, utilizing the Pixy Camera's libpixyusb linux libraries.

The primary obstacle we encountered with the Pixy Camera was its dependence on color based filtering algorithms to detect and track objects of interest. The Pixy cam learns what the objects of interest are for a given system by analyzing example objects presented to it in its “teaching” mode.

This is known as a “supervised” object recognition system, since a human user is indicating objects of interest in the environment by “tagging” them (color tagging in the Pixy cam’s case). Our computer vision system was an “unsupervised” case, i.e. the vision system had to be capable of detecting and tracking “untagged” objects of interest. Nothing in the Pixy cam’s existing source code allowed for general motion tracking (tracking the movement of an “untagged” human body), so the Pixy Camera is treated as a simple capture device in our system (albeit one with effective exposure correction algorithms and several resolution options). All the “heavy lifting” of our computer vision software is thus carried out onboard the Raspberry Pi 3.

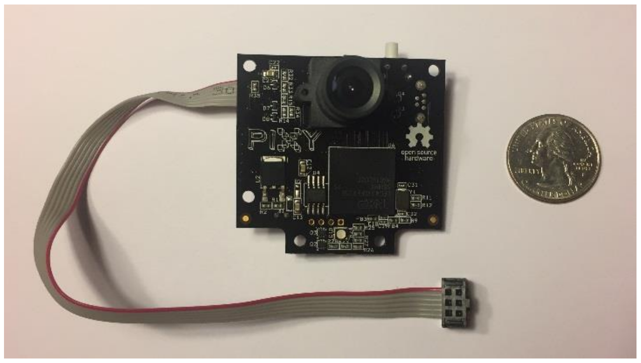
Fig. 4 Pixy Cam or cmuCam5.

B. *Motion Segmentation*

Motion segmentation is one of the core fundamentals of object tracking in computer vision. It consists of separating the objects being tracked from the background environment. There are countless algorithms and methods by which real-time motion segmentation is performed, however one of the simplest and most effective methods is the concept of image difference.

Image difference seeks to construct a map of differences between two contiguous frames from a video snippet. It identifies these differences by comparing the intensities of each of the consecutive frames corresponding pixels. This technique is extremely sensitive to image noise (random variations in brightness and color that can be thought of as “digital background noise”) and is practically unusable when the camera capturing the images is moving. Due to the fact that the installation utilizes a fixed camera solution and does not require fine-grained object recognition, this primitive but effective technique was a viable solution. Image difference is best at tracking blobs of motion moving through a series of frames.

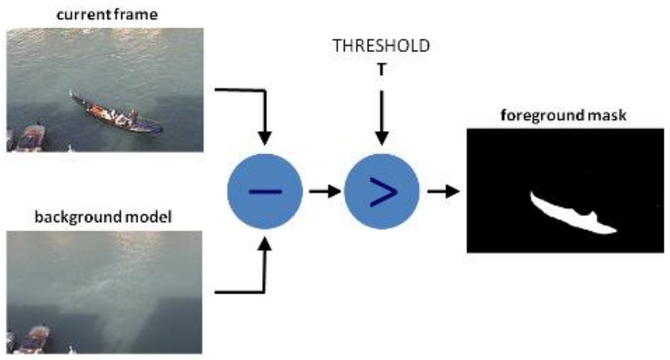

Fig. 5 Image difference visualization, reprinted with permission from OpenCV.org.

Typically these motion blobs are then used as input to more advanced computer vision techniques, which might attempt to identify specific objects or produce a more fine-tuned structure for the objects of interest. However in the case of the interactive waterfall, observers can very well be thought of as vague regions of movement, keeping in mind that the resolution of the graphical waterfall responding to the observer movement is made up of only 16 solenoids. To reduce the rate at which the Pixy Cam observes a “false positive” (instructs a solenoid to turn active when no observer is present in front of the solenoid) a double thresholding solution was utilized.

C. *Hysteresis Thresholding*

The “difference image” is stored as a single black and white .pgm file. Double thresholding is then carried out to reduce foreground interference. “Double thresholding” or Hysteresis thresholding is used to reduce image noise and eliminate any change between the frames that is not the result of observer motion, such as movement in the background of the observed scene or brief changes in brightness.

The thresholding is carried out first by applying a high threshold value to all pixel intensities in the difference image. This is the “first pass”. Any pixel that is above this initial high threshold is set to an intensity of 255 (white) and marked as a “peak”. Any pixel intensity that falls below this initial threshold is passed over for now. For the algorithm's second pass all pixels that passed the initial high threshold (the peaks) are re-examined. Any pixel that is adjacent to a peak pixel and has intensity greater than a second low threshold value is now additionally set to 255 (white) as well. This method works due to the observation that pixels adjacent to pixels with a high intensity are likely to be a part of the same region of motion.

It is important to recall that this thresholding is being performed on the difference image, hence regions being marked white in this final image are the regions where motion is occurring in the frame. With the final image computed, further calculation can be performed to determine what data to send to the solenoid control unit and LED switching circuit. The pixel intensities of the thresholded difference image are converted to values which are stored in a two dimensional array. This two dimensional array can be thought of as a dataset where every occurrence of the unsigned char value 255 represents the x and y coordinate of a pixel in the captured frame where a region of motion has been observed. A graphical example of hysteresis thresholding in action is presented in Figure 5.

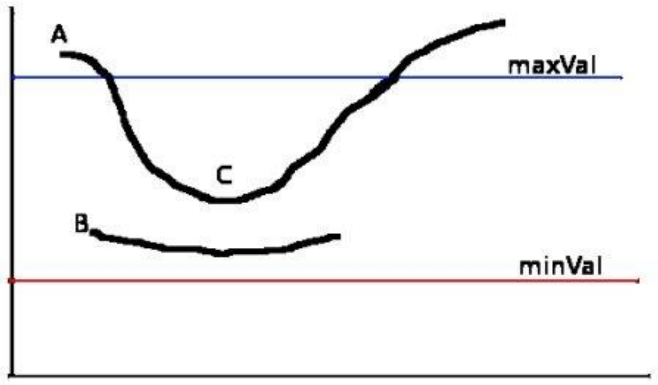

Fig. 5 Hysteresis thresholding example, reprinted with permission from OpenCV.org.

D. *Low Threshold Calculation*

Over the course of testing our installation in the motion interaction mode we found that the most important factor in reducing image noise and background motion interference was the constant which is used as the low threshold during the Hysteresis thresholding process.

Rather than selecting a static value which worked well during our limited testing of the installation, the team decided to take a more dynamic approach to low threshold calculation. When the system is starting up in its motion interaction mode there is a period of five seconds during which it performs a “background analysis”. This analysis consists of capturing 10 frames of the background scene (2 per elapsed second). The system then performs arithmetic image difference between each frame and its successive frame, so that it ends up with five two-dimensional image difference arrays. The average change in pixel intensity is calculated for each two dimensional array, creating five numerical values representing the average change in pixel intensity for a given second. These five values are then averaged to arrive at a final numerical value which is the dynamically determined low threshold value.

IV. Spectrum Analysis

For the Audio Interaction mode of the installation, a USB sound card is connected to the raspberry Pi 3 which serves as our System Control Unit. We then utilize a modified version of existing open source Raspberry Pi software known as Lightshow Pi to interpret our audio input into output signals. Our modifications for this Python script involved converting its output signals into a format which would reproduce aesthetically pleasing patterns on our graphical solenoid array.

V. Output Control Software

The outputs of the installation consist of the graphical water solenoid array and the variable brightness LED strip which provides colored lighting for the waterfall. The solenoid switching array is controlled by a serial data line from the MSP430G2553 or Solenoid Control Unit. The LED strip however, is controlled directly by signals from the Raspberry Pi 3 or System Control Unit.

A. *Solenoid Control*

The graphical solenoid array’s output signals begin as a two dimensional array representing the results of our image difference calculations. A custom algorithm loops through this array and determines the longest consecutive region of motion in the x axis, or the longest series of consecutive pixels which have been marked as a region of motion in the most recent frame. The starting position as a ratio of the captured frame and the width of the horizontal region of potion in pixels is stored off by the Raspberry Pi 3.

The Raspberry Pi 3 then transmits this data to the MSP430G2553 or Solenoid Control Unit which scales the region of motion’s starting x pixel coordinate and total pixel width to the resolution of the graphical waterfall array. Once the necessary signals for each solenoid have been determined, existing open source Energia shift register functions are used to send the output signals to the PCB.

B. *LED Strip Control*

The output signals for the LED strip are determined in a similar method, as the two dimensional image difference array is looped through and the longest consecutive region of motion in the y axis is attained. The values for the starting pixel of the region of motion and the vertical length of the region of motion are stored off by the Raspberry Pi 3. These values are then used in later function to change the brightness of the LED strip through Pulse Width Modulation.

VI. Power Systems

A. *Power Prototyping Restrictions*

Our project started with extremely ambitious power design objectives. But due to customer concerns, budget and time constraints some aspects of the design were scaled back. Originally there were supposed to be several solar panels with an inverter built from a developer kit, this alone would cost close to $1000 dollars. Our client OUC also voiced concerns that our inverter design would not meet NEC regulations so would not be useful in their final design. OUC also wanted to use power from the grid to power the final design for the sake of simplicity and to eliminate the need for batteries. This combination of factors led us to drop the inverter and solar panels from our project and to focus more resources towards improving the aesthetic display.

Another item that proved to be too difficult to execute was a high power rectifier. There was two options for this which included a more amateur design that had high energy loss and a more professional design that was composed of about 140 components. The high complexity design couldn't be completed due to the large amount of soldering and difficulty trouble shooting such a highly complicated system. The simpler design used a large specialty transformer costing $100 and had a very long lead time. The cost of this design was fairly high, $250, because of several factors. The PCB would need very thick copper traces to handle the high current which increases the price. The high current also makes using an integrated chip for the H Bridge impossible so one would have to be made out of expensive high power diodes. Finally to boost the current high power BJTs would need to be used. The cost of this rectifier would greatly affect the overall budget for our project. Because of these factors a commercial rectifier used for medium to large scale LED displays was used.

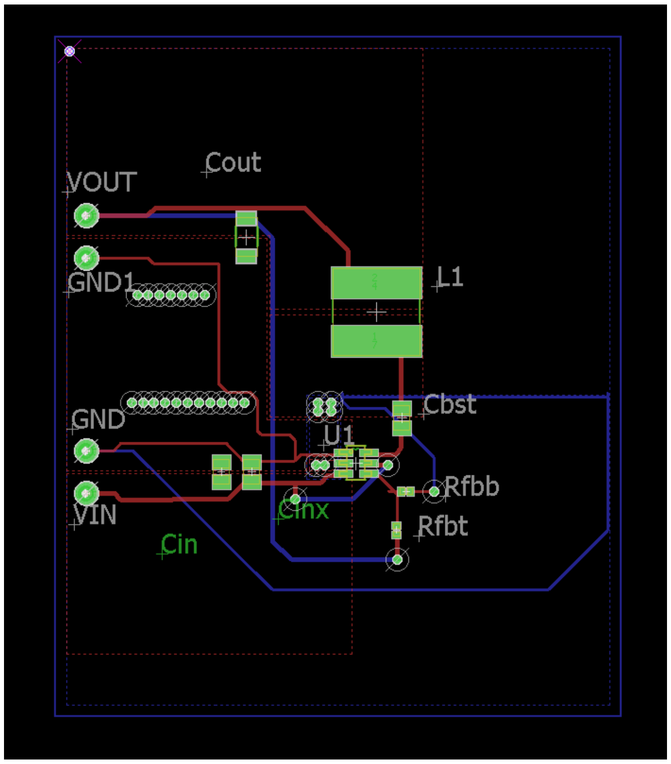

Fig. 6 Above is the PCB layout for both of our voltage switching circuits. However the values of the passive components and the ICs used in the two circuits are different.

B. *Printed Circuit Board Design*

The design of the printed circuit board was constructed using the schematic software EagleCAD. The PCB contains 4 main components: the solenoid switching circuit, the shift register, the MSP430 microcontroller, and the power supply rails. The solenoid switching schematic consists of a N-channel MOSFET, with a 1N4001 diode and the solenoid in parallel. These components all share a common 12V rail to supply power to the solenoid. The switching of these solenoids are controlled by the output from the two 8-bit shift registers and there connection to the MOSFETs. These MOSFETs act as switches to regulate the 12 volts supplied to the solenoid, when the shift register outputs high to the gate on the MOSFET the 12V is able to turn on the solenoid. The diode is placed in parallel with the solenoid in order to prevent fly back voltage to the MOSFET when the solenoid induction current is dissipated. The two 8-bit shift registers are daisy chained together by connecting pin 9 to pin 14 as seen in Figure 7.

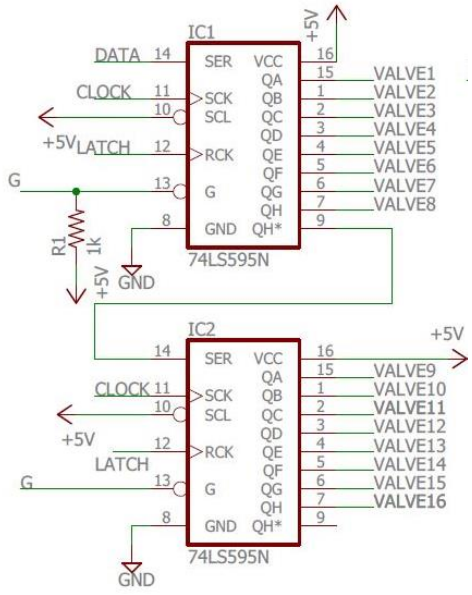

Fig. 7 Daisy Chained 8-bit Shift Registers.

This allows the MSP430 microcontroller to be able to communicate to all 16 solenoids simultaneously. The MSP430 is mounted on the circuit board with a 20-pin socket and receives input commands from the Raspberry Pi. These external connections are made through female header pins mounted on the outside edge of the PCB. These data lines allow for transmission of information from the sensor camera through the processing components out into the selector lines controlling the state of the solenoids. All these components mounted on the printed circuit board share a common 12V, 5V, or 3.3V rail. The 12V rail is a direct connection from an AC to DC power supply with 20A, necessary for the large power draw from the solenoid array and LED lights. These 12 volts are then regulated down to 5V and 3.3V using external voltage regulating PCBs. The 12V is used to power the solenoids and water pump, while the 5V regulator is used to supply the power needed for the Raspberry Pi, CMUcam5, and Audio input sensor. Further voltage regulation down to 3.3V is necessary for powering the MSP430 microcontroller.

After completion of the final schematic design in EagleCAD, the circuit is ready to be transferred to the physical printed circuit board layout design. Considering the high functionality and power draw of the solenoid switching circuit consideration for heat dissipation were evaluated. Placing the MOSFETs in parallel position along the length of the circuit board allowed to optimal cooling and preventative over heating measures. The control components, MSP430 and shift registers, are placed in the middle of the PCB because of their centralized connection to all the components on the board. This centralization allowed for minimal wiring layers and vias need when printing the board. The outer connections to the solenoid screw terminals are all placed in a row of 16 in order to consolidate the wires need between the PCB and the solenoids. The input and output pins used for communication with the other controllers are placed on the exterior of the PCB to allow for ease of access when debugging component interconnections. The layout of the final printed circuit board is shown in Figure 8, notice the use of 45 degree angles to minimize the wiring needed.

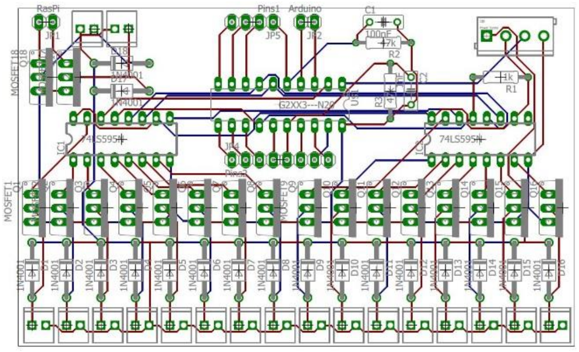

Fig. 8 Printed Circuit Board for Solenoid control

The final printed circuit board was constructed by the company Elecrow, and was shipped with additional PCBs for troubleshooting. All the components were through hole mounted and soldered on by the members of the design team.

VII. The Engineers

Ben King grew up and attended high school in Olathe, KS before moving to Florida. He transferred to University of Central Florida in 2015 with an AA degree from State College of Florida, and is graduating in May 2017 with a Bachelors in Electrical Engineering. His interests include microelectronics and power delivery. He hopes to work designing and testing electronic circuits for interesting gadgets and various applications.

Tahte Perkins: started at the University of Central Florida in 2011 after graduating from Olympia High School in Orlando, FL. His interest in the field of electronics began when he started building his own computers and started writing simple programs in high school. In college he went into electrical engineering and interned with Conam Construction in Anchorage, AK summer of 2013 and 2014. There he was able to increase his real world understanding of high voltage power systems and large scale and small.

Jack Gray from Rockville, MD began his interest in engineering as an International Baccalaureate student at Seminole High School where he gained an understanding of analytical thinking and an appreciate for the sciences. He continued his pursuit of knowledge at the University of Central Florida in 2013 towards a Bachelor's of Science in Electrical Engineering. After being a student-athlete for four years on the water polo team, Jack hopes to continue his passion for inquisition and analytics through systems engineering.

Connor Heckman began his education in computer engineering at the University of Central Florida in the fall of 2012. His interests include image processing, real-time embedded systems, and machine learning. Throughout his final semesters at UCF, Connor has worked as a College student technician in the Lockheed Martin/UCF CWEP Program. Following his graduation, Connor will be starting full time at Lockheed Martin Missiles and Fire Control as a Software Engineering Associate on the LRASM program.

VIII. Acknowledgements

The authors are grateful for the support and input of the Orlando Utilities Commission throughout the course of this project.