Refining LEGOLAS
CS496 Capstone Project

Brendan Clark, Audrey Versteegen

2024-05-08

Contents

Revision History

added points to user stories, used "the system shall" format for the user stories.

1 Introduction

1.1 Project Summary

Our client Dr. Mary Lowe is a physics professor who obtained a LEGOLAS robot from a workshop at the University of Maryland. LEGOLAS, the LEGO-based low cost Autonomous Scientist, is a robot designed and created by students to be able to mix two compounds and analyze their reactions to each other. The process of mixing and analyzing these compounds by hand is laborious, inefficient, and time consuming. The aim of the robot is to automate this process to save time. The robot uses a machine learning algorithm to determine the quantity of each substances to mix. Afterwards, the robot measures the pH level of the mixture. The problem our client faces with this robot is that its pH sensor is often inaccurate in its positioning, which leads to ruined data and time wasted. Our goal is to fix the inaccuracy of the pH sensor by implementing a camera and utilizing computer vision to allow LEGOLAS to make adjustments in real time to the positioning of the pH sensor.

1.2 User Classes

1.2.1 Researcher

Those researching the physical sciences can use LEGOLAS to automate the more tedious aspects of it. The will have experience mixing and testing substances, and the LEGOLAS machine must hold up to the standard. There is no security required for this product.

1.3 Risk Analysis

The following risks have been determined may affect this software's completion can be seen in Table 1.

Risk	Impact	Priority	Overall Rating	Mitigation
Some component of the LEGOLAS robot breaks.	5	4	20	1. Food and Beverages will not be allowed on the same surface as the LEGOLAS robot in order to minimize electronic malfunctions due to messes. 2. When running, the LEGOLAS robot will be under constant supervision. 3. Our client, Dr. Lowe has mentioned the department being potentially able to supply parts. If dire, they may be contacted for support.
The department at the University of Maryland requests the return of the LEGOLAS robot.	5	2	10	1. Communication with both the client and the UMD department to ensure that the device is continued to be allowed to be in our possession. 2. Should this come to be, development on the camera and the computer vision
The wire attached to the camera isn't long enough to position in an optimal place.	3	1	3	1. Additional wire extensions can be bought as needed should this occur.

Table 1: Risks to the software.

2 Requirements

This section outlines all requirements for the software, primarily via user stories. If epics are used, they are broken down into user stories, and large user stories are broken into tasks. Non-functional requirements are included separately.

2.1 Functional User Stories and Tasks

This section lists out all user stories, organized by user type. Within each user type, stories are organized by priority, lowest to highest.

2.1.1 The LEGOLAS System

ID: pH Sensor Miss Notification (N1)

1. Priority: low
2. Points: 2
3. **Identify when the pH sensor has missed a mixing well**
4. **So that the user can reset the system to be able to continue to collect data**
5. **Definition of Done: the LEGOLAS system has some sort of notification or alarm to notify the researcher that there was an error in collecting data**
6. Tasks:
   (a) N1T1: Identify Miss. Detect when the system has missed its target.
   (b) N1T2: Notify User. When a miss occurs, emit a sound and display an error.

ID: Install Camera (C1)

1. Priority: High
2. Points: 2
3. **The system shall have a camera installed**
4. **So the system can use computer vision to assist in the positioning of the pH sensor**
5. **Definition of Done: the LEGOLAS system has a camera installed**

6. Tasks:

   (a) C1T1: Position Camera. Find a suitable position for the camera to be placed.

   (b) C1T1: Connect to Raspberry Pi. Connect the camera to the Raspberry Pi and make sure its recognized.

ID: Computer Vision (C2)

1. Priority: High

2. Point: 21

3. **The system needs the pH sensor to be accurate in its positioning**

4. **So that it can efficiently collect data**

5. **Definition of Done: the LEGOLAS system correctly uses computer vision to identity where the pH sensor needs to be positioned to be able to take measurements.**

6. Tasks:

   (a) C2T1: Connecting Camera. Connect the camera to the existing software.

   (b) C2T2: Identify Dot. Make sure that the camera can identify the positioning dot.

   (c) C2T3: Identify Mixing Well. The installed camera needs to correctly identify the mixing wells.

   (d) C2T4: Calculate New Position. Based off of the identified mixing well, calculate the changes needed to adjust the pH sensor's position.

   (e) C2T5: Reposition pH Sensor. The system will need to make adjustments based off of the calculated changes to the pH sensor's position to ensure that it does not miss the well.

(f) C2T6: Take pictures for the dataset. We will need to take a variety of pictures to train our model for identifying the mixing wells.

(g) C2T7: Label the dataset.

(h) C2T8: Normalize radial distortion.

2.2 Nonfunctional Requirements

2.2.1 Category: Performance

ID: Speed and Accuracy

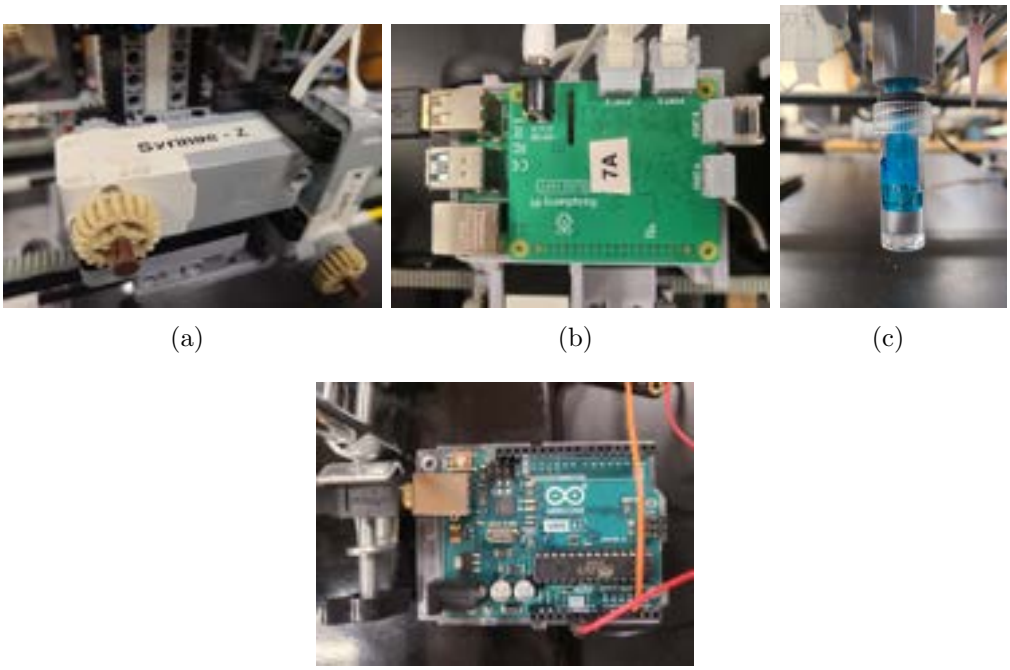
1. Description: When it comes to having the LEGOLAS machine maintain and improve its functionality, it is important to keep in mind the speed at which the decisions and calculations are made. While accuracy is the main goal of this project, it should not be at the major expense of speed. The accuracy of the camera in identifying the location of the mixing well should be as reliable as possible.

2. Affects: Ensure that LEGOLAS chooses its positioning within a reasonable amount of time. No more than 10 seconds. While the ideal result is 100% accuracy and precision, a result of over 90% correct identification of the tray is important for the reliability of the machine as a whole.

3 Design

3.1 Hardware

The LEGOLAS robot was given in full possession of the needed hardware to address it's task. In addition to the infrastructure built from Lego pieces and the aluminum frame, the mechanics are as follows:



(a) (b) (c)

(d)

Figure 1: (a) (b) Raspberry Pi with BuildHat (c) pH sensor and Pipette (d) Arduino

1. Five Lego Motors: These are responsible for all movement, from positioning the system's pH sensor and pipette, lowering them, and emptying and filling the pipette.

2. Two Raspberry Pis: Executes code responsible for controlling the motors and communicates via RPyC server with the software ran on our computers locally. The one in Figure 1 also has attached a BuildHat.

3. One pH sensor: The analog pH sensor is responsible for measuring the pH level of mixtures. This is the blue sensor in Figure 1.

4. One Arduino Board: collects data from the pH sensor.

5. One pipette: Responsible for transporting the materials to each tray well. This is the pink tip in.

Throughout the implementation and development of computer vision in the LEGOLAS machine, there are other pieces of hardware that will either make their way into the system,

(a) (b) (c)

Figure 2: (a) Arducam Camera (b) Raspberry Pi (c) 12 Inch Ribbon Cable

or be used in the preparation of the identification model. They are:

1. A Raspberry Pi: As seen in Figure 2, this will be used to train the model in identifying the locations of the tray wells. The camera, mentioned next, will be plugged into this Pi. Should the model prove complex and demanding enough to merit it's own Raspberry Pi for computation, this will also be incorporated into LEGOLAS.

2. Arducam Camera: Main component for computer vision and will assist software in fine tuning the positioning of the pH sensor. Detailed in Figure 2.

3. Ribbon Cables: Ribbon cables are what connect the Raspberry Pi and the camera. We used a twelve inch cable as shown in Figure 2.

The Raspberry Pi was initially intended for use in gaining familiarity with the hardware of Raspberry Pis. However, it proved itself to be more up to date than those on the LEGOLAS machine. As such when the original Raspberry Pis were unable to download needed packages, one was replaced by the newer Raspberry Pi. The Arducam camera is mounted on the

3.2 Software

3.2.1 Software Overview

Figure 3 details a simplified sequence diagram for the LEGOLAS machine. The functionality and logic were already fully programmed before our research started. This includes the

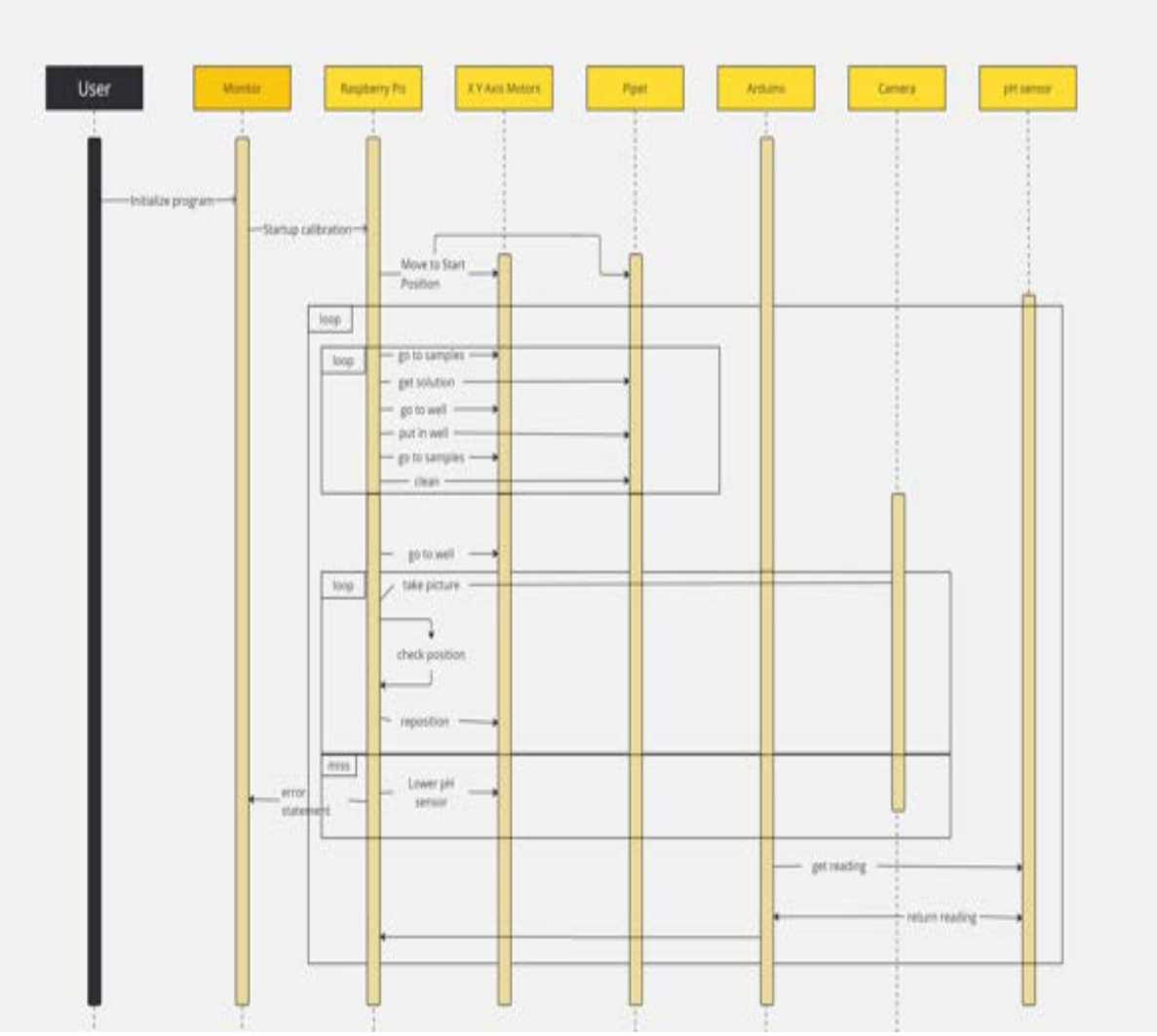


Figure 3: LEGOLAS Sequence Diagram

logic for the movement on each axis, the collection and distribution of the solutions, the cleaning of the pipette before changing solutions, testing the pH of the solution, assessing the information using machine learning and more.

As seen in the lower half of Figure 3, the assessment of the pH sensor's position is the main focus of the project. Once the assessment it don and the calculations for adjustment are made, the location will be assessed once again, until no adjustments are needed, or are needed to non significant amounts. Should the sensor miss, the error window will pop up for the user and immediately exit, saving the information gathered. Here are a couple of the tools we plan on utilizing as the model is incorporated:

1. RPyC server: responsible for communication between the software running on our computers and the raspberry pi. Allows for execution for the code on our computers to be executed on the raspberry pi. The server is also responsible for sending the images needed for the model.

2. OpenCV: potentially the library that we will use in order to identify the mixing wells. We might have to create and train our own model to be able to identify the wells if we are not able to use or fine tune any of OpenCV's pre-trained models to our desires.

3.2.2 Interface Design

As a robotic programming assignment, there are not many User Interface elements present in this research. Figure 4 describes the UI elements already in place in the LEGOLAS system. Figure 4 displays the Manual Mode to control the mechanics of the LEGOLAS machine. Calibration windows allow for initial positioning of the tray wells, and others allow for the control of the fluids being transported by the pipette and other settings.

Figure 4: LEGOLAS Screen Flow Diagram

Figure 5: LEGOLAS error

Figure 5 is the design for the error message that we have implemented. The screen will appear to the user when the system detects that the pH sensor has missed its target.

3.2.3 Programming Languages

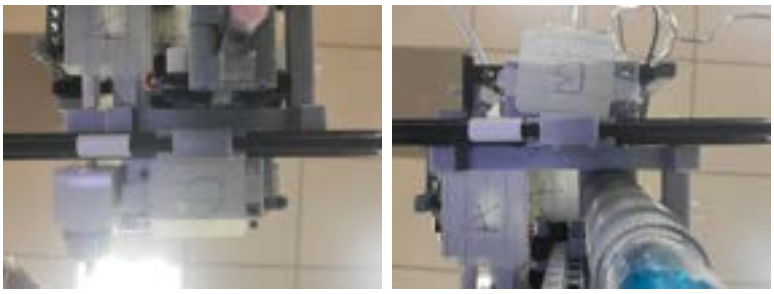
1. This project will be written using python. This will mostly take place in PY files, however there are a few notebooks written as well.

4 Approach

4.1 Methodology

4.1.1 Camera Installation

The camera needs to be positioned on the independently moving segment of the LEGOLAS machine in order to avoid complications and snagging the wire connecting the ArduCam camera to the raspberry Pi. Four possible locations were found, and labeled A, B, C, and D. Position A, as shown in 6 is closest to the tray, but distant from the pH sensor, making



(a) (b)

Figure 6: (a) Locations A, C, and D (b) Locations A, B, and D

the angle of the wells more severe. Positions B and C were located on either side of the LEGOLAS component. Position D is located alongside the pH sensor, but is higher up than the other potential positions.

The placement of the camera must allow for a clear line of sight towards the tray wells closest to the pH sensor. As such, B and C were disregarded. Due to their placement on the supporting sides of the machine, their view was not reflective of the pH sensor's position. Location A was inline with the pH sensor on an axis, and allowed a somewhat clear view of relevant area. However, position D provided the closest perspective of the pH sensor's location and while the position is further away, this added distance did not negatively effect the images taken.

4.1.2 Data Collection

Images are taken using the ArduCam camera in position D. Because of the pH sensor's limits in reaching the table, two trays are stacked on top of each other and are placed within the sight of the camera. A series of nine images can be taken using a script to take pictures every five seconds and a flashlight to introduce variety in the lighting. This process is repeated, moving the trays around the area, taking care to include edge cases. In order to avoid training to one surface, other backgrounds need to be included in the dataset. The image collection process is repeated using a slab of wood and then a piece of paper so that wood and white backgrounds are taken into consideration in the model.

This collection of images are then duplicated and altered. Through a combination of rotation and hue variation, The database has 1,728 images. These are then cropped down to the area of interest: the area in front of and underneath the pH sensor.

Once the images are taken and processed, they are uploaded to the workspace in CVAT, a free and open-sourced tool for labeling images. Each image is labeled using the ellipse tool to outlining each well. In order to minimize error in human variation, a circle was drawn and then duplicated throughout the image in order for the circles to remain as uniform as possible. The propagate tool allows for continuity throughout a series of images, and is used whenever applicable in order to increase consistency.

4.1.3 Faster R-CNN Development

Faster Region-based Convolutional Neural Networks(F R-CNN) use trained models to identify instances of a class throughout an image. Regions of interest are identified, then passed through the model to classify it. By using the ResNet50 pretrained weights, the model will be able to quickly train for identifying the location of the wells.

4.1.4 Incorporation with LEGOLAS

Once the model is trained it can be incorporated into the LEGOLAS machine. The trained model is shared. The LEGOLAS machine operates as normal up until the point where the pH sensor would be extended into a well. At that point, the camera takes a picture. This picture is then run through preprocessing, and then fed into the model's trained weights. Each bounding box can be simplified into a center coordinate. These coordinates are then grouped up into neighboring points, then according to their x and coordinate similarity. These x and y groups create the predicted grid of well centers. Calculations based on the positioning of the grid relative to the pH sensor determines if adjustment is necessary. If so, the adjustment is made, and another picture taken to verify the placement. This loop continues until no adjustments are needed.

4.2 Tools

The Computer Vision Annotation Tool(CVAT) was used to label the images. CVAT is online, and allows for collaboration, which decreased the difficulty in labeling the dataset by hand. Sobolsoft's XML splitting tool was used to prepare the data for training. As this project uses python, libraries including Numpy, Tensorflow, SKLearn, JSON, Glob, CV2 and more were used in the development of the scripts.

5 Experimental Design

5.1 Data Collection and Manipulation

The camera was positioned in location D, shown in Figure 6 and secured temporarily using toothpicks for support and tape for adhesion. Because the pH sensor is not able to reach the bottom of a well when the tray is on the table, it is required that a second tray be placed underneath the first so that accurate pictures could be taken. Using a python script, images are collected in batches of nine. Within a batch, the position of the tray relative to the camera remains the same, as well as the background. The script allows for a five second wait between capturing a picture, in which the lighting of the area can be changed. This was done using the flashlight feature of a cell phone. Between batches, the position of the tray is altered. This allows for the coverage of the end cases, where the intended well is on the edge of the tray, as well as instances where the well is surrounded. Once a background had been thoroughly documented, it was changed. In addition to the black tabletop and using an additional tray for height, pictures are also taken with the tray on a wooden block, as well as on a sheet of paper to replicate the variety of surfaces that a LEGOLAS robot may operate on.

Once this dataset is acquired, alter the images by increasing the hues of reds, greens, and blues in order to get additional variation. Rotate these imaged by 15 degrees clockwise and

counter clockwise in order to anticipate a tray that is not perfectly aligned. These images are then cropped in order to isolate the area of interest. This area being the square directly in front of the pH sensor.

This dataset is then loaded into CVAT, where each image is carefully labeled. When exported, the entire dataset is written onto one XML file. These are split using Sobolsoft's Split XML Into Multiple Files Software. To correct the naming discontinuity as well as the adjustment from documented circles to documented squares, the 'read and rename' python script is run. This converts the documentation of cx, cy, rx, ry into the values of xmin, ymin, xmax, and ymax and naming the file according to the picture it labels. Once all files are accounted for, the dataset is finished.

5.2 Faster R-CNN Model

The database directory is read in. In this scenario, the directory path of the images is '/content/drive/MyDrive/total' and '/content/drive/MyDrive/LEGOLAS split rewrite' respectively. The annotation files are verified to have at least one bounding box written, with any exceptions and the correlating image being removed. A random twenty percent of the annotation and image pairs are moved into a 'test annotations' and 'test images' folders. This is to establish a testing set to verify the accuracy of the model at a later time.

Functions are defined to read the XML annotation files and images together, recognise the bounding boxes and handling the transform, which converts the images to a tensor.

The model is then defined using the torchvision fastercnn resnet50 fpn() function. The model is pretrained and starts with the ResNet50 weights. The box predictor for the model is then established as a Faster RCNN box predictor. Due to limitations, the model is trained in batches of 20 images over 20 epochs. More epoch were tested, but the loss plateaued around the 20th cycle.

5.3 Integration with LEGOLAS

The main concern with the model's integration into the LEGOLAS machine is that it will have a negative effect on performance. While the model is trained to help navigation, it is possible that the cost of processing is high, or that the model will be indecisive about location and directions. There will be control data collected, recording how long each process takes and of the frequency of failure. This information will be contrasted with the data collected from the LEGOLAS machine when working with the model.

Integrating the model into the LEGOLAS system required loading the model into the program before usage. The current implementation of the model is that the LEGOLAS will move to a position, take a photo of the current position and send the photo to the computer running the program issuing commands to the LEGOLAS machine. The program will then convert the image bytes received from the Raspberry Pi over the rpyc server and then feed the image into the model. The model then pre-processes the image in order to apply the model. Then, the program can make predictions as to where the center of the wells are. The model returns a list of bounding boxes, and the program converts them to a single coordinate representing the center of the circle. This is done by averaging the min and max values of x and y.

The next function to be called searches for neighbors for each point. It will then draw horizontal and vertical lines intersecting all the dots that label the center of a well. Using these horizontal and vertical lines the program can then make adjustments to the position of the pH sensor. This is done by taking the pH sensor's current position, a fixed location in the image a [394, 663], and checking its relative position to the horizontal and vertical lines to make sure that the sensor is aligned properly. The program then determines whether the pH sensor is aligned or not, if it is aligned nothing will happen, if it is unaligned the program will make a calculation to change the pH sensor's position and issue movement commands to the LEGOLAS system to adjust its position so that the pH sensor aligns over the well.

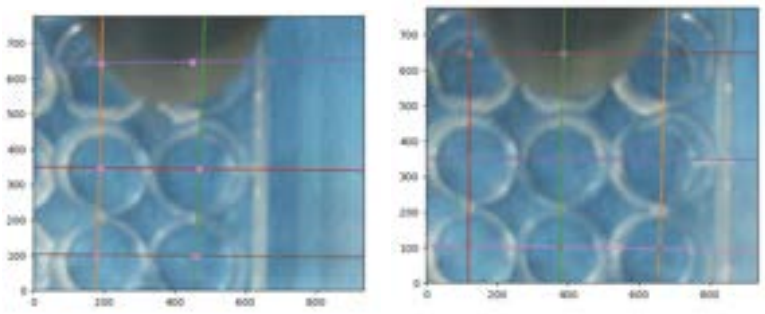
6 What is Completed

For our functional requirements we have completely implemented pH sensor miss notification (story N1) and installed the camera into the system (story C1). For implementing the computer vision (story C2) into the system we have completed various sub tasks. This includes taking pictures for a dataset, labeling the dataset, and having training a model for identifying the wells using computer vision as well as implementing code that deals with the radial distortion from the camera. We have also implemented a way for the raspberry pi to send images to the main program through an rpyc server. These images are then used for the model that we currently have implemented into the existing LEGOLAS system. The current implementation is as follows. The LEGOLAS system will move to a position, it will then take a photo of its current position after it has moved. It sends this photo to the computer running the program issuing commands to the LEGOLAS system and then feeds it to the model which applies various pre-processing effects to the image in preparation for its predictions. The model then makes predictions on where the center of the wells are and places a dot on each of them. The model then places horizontal and vertical lines that intersect the center of the wells. The program then uses the position of the pH sensor in relation to these vertical and horizontal lines to determine whether the pH sensor is properly aligned. If it is, it will do nothing, if the sensor is not aligned, it will make a calculation to change the position of the pH sensor and then adjust the position of the pH sensor.

7 Results

The Faster Region-based CNN model was trained on the gathered dataset somewhat successfully. In Figure 7 the loss value over the epochs is shown. As the graph shows, loss in the first epoch had a value of around 16, then as the model trained, settled around the eight mark. The ideal value for a model is zero, so while eight is not a large number, there is room for growth. We were comfortable training the model up to the 20th epoch because of the

Figure 7: Total loss in each epoch as model trained



(a) (b)

Figure 8: Points and grid predicted by the model

size of our dataset. The model itself did not test very well, with a precision score of 41%. We had hoped to achieve closer to 90% precision.

Having given the statistical concerns of the model, it should be mentioned, that it performs well in application. Figure 8 shows two examples of the models predictions. Once the model has predicted the image's boundary boxes, the predictions are processed and grouped and lines are drawn. These images show that the model does work on the high contrast backgrounds. It is believed that the images with the white backgrounds may have skewed the statistics.

That being said, the calculations drawn from the lines are not consistent. With few exceptions, the points are plotted correctly and the vertical and horizontal lines are accurate. When these are processed by the program, and the locations assessed, there is a disconnect between the projected distance and the responding motion. This may be due to a few

factors, one being a miscalculated ratio between the pixels from the image to the imaginary grid LEGOLAS observes. Another possibility is faulty logic.

In terms of efficiency, the process of predicting with the model, grouping the points, graphing the lines, and calculating the needed motion is not inefficient. These calculation take approximately five seconds per iteration, which is a number that we are happy with. In comparison to the normal CNN model who would take roughly five minutes to process an image.

8 Reflections on the Project

After having worked on this project throughout the semester, we believe the most critical part of the software development process has to be planning. From the beginning if the design and methods are well throughout and established it makes the development process significantly easier. Of the functional requirements that we have, we stated that the implementation of computer vision into the LEGOLAS system would be 13 points. This was clearly an underestimation and was significantly more complex than we anticipated. If we were to re-estimate the points for the story, we would say that it is worth 21 points, possibly more. The same could be said about installing the camera as it turned out to be much more of a hassle than anticipated because of the hardware. Instead of 2 points it should be worth 3. These underestimations are because of our inexperience in working with computer vision technology and raspberry pis.

The requirements for our project were straightforward for the most part and did not change. The core of the project was implementing computer vision and while it could be split into many tasks, we felt like none of them warranted being complete user stories as they do not completely capture a complete user case scenario. These tasks that this user story was split into did change throughout the semester as we learned about new technologies and produced new ideas on how to implement computer vision. Such as having to learn

how to deal with the radial distortion the camera for the system had. This then created an additional task where we needed to implement a way to normalize the radial distortion.

Working with a real customer was exciting and motivating as there was a real desire for the product that we were trying to develop. Throughout the software design process, we learned how important diagrams and figures can be when conveying what the goals and how they will be achieved are. From our document we utilized a flow diagram for the user interface, and we used a sequence diagram to describe how the software will work. The presentations made me realize how important figures can be as well.

In regards to the software, we have learned more about how machine learning works and about the process in training a model. Learning something completely foreign is both exciting and frustrating as on the one hand we get to explore a new technology but on the other hand we are inexperienced. As such, when we get stuck on an issue for a long time without being able to find a solution it makes it more frustrating to deal with. In addition to learning about how to train a model using machine learning we had to get familiar with the raspberry pi environment and its integration with the LEGOLAS system. This involved familiarizing ourselves with how the system works and its code-base. Up until now, every assignment or project that we have had has been built from the beginning. So, to be thrown onto an already existing software is a bit challenging as we do not know every aspect to the software, so there was a bit of a learning curve trying to figure out someone else's code.