Project Report on M&M Sorting Machine

Designed by:

Archit Tatwawadi

Mohak Patel

Project Description/Specifications:

- M&M Sorting Machine was developed in this project by implementing a neural network for color detection on FPGA.
- Nexys4 DDR board was the main controller of the system which was used to implement the color detection part and controlling the remaining hardware boards.
- Raspberry Pi was used to integrate the camera with our system.
- The Raspberry Pi controlled the camera and clicked the images of M&M and processed the image to be sent to Nexys4 board via UART. The Nexys4 board initiated the UART communication with the Raspberry Pi.
- The Raspberry Pi implemented an object detection mechanism to detect the M&M. Then the color portion of the M&M image was cropped and converted to a 28x28 pixels grayscale image.
- This image was then sent via UART to the Nexys4 board.
- The 784 pixels of the image received by the UART was stored in an array. This array was passed on to a neural network implemented in C language for further detection of color of the image.
- After detection of the color, the detected color is then displayed on the OLEDRGB display and the servo motors attached to the hardware are correspondingly moved to drop the M&M in the detected colored bowl.
- The user has to put another M&M on to the device and press the Center button on Nexys4 for further processing of M&M’s.

IP Blocks used:

1. 3 AXI Timers
   AXI Timer 0 – For Nexys4IO module’s clock.
   AXI Timer 1 – For PWM Generate mode.
   AXI Timer 3 – For PWM Generate mode.
2. PmodOLEDrgb display.
3. 2 UARTLITE IP
   UARTLITE 0 **-** For JTAG purpose, Baud rate 115200, 8 bits, no parity
   UARTLITE 1 **-** For Communication between Raspberry Pi and Nexys4, Baud rate 115200, 8 bits, no parity
4. Memory Interface Generator
5. Nexys4 IO module
6. Microblaze.

Block Diagram of the System:

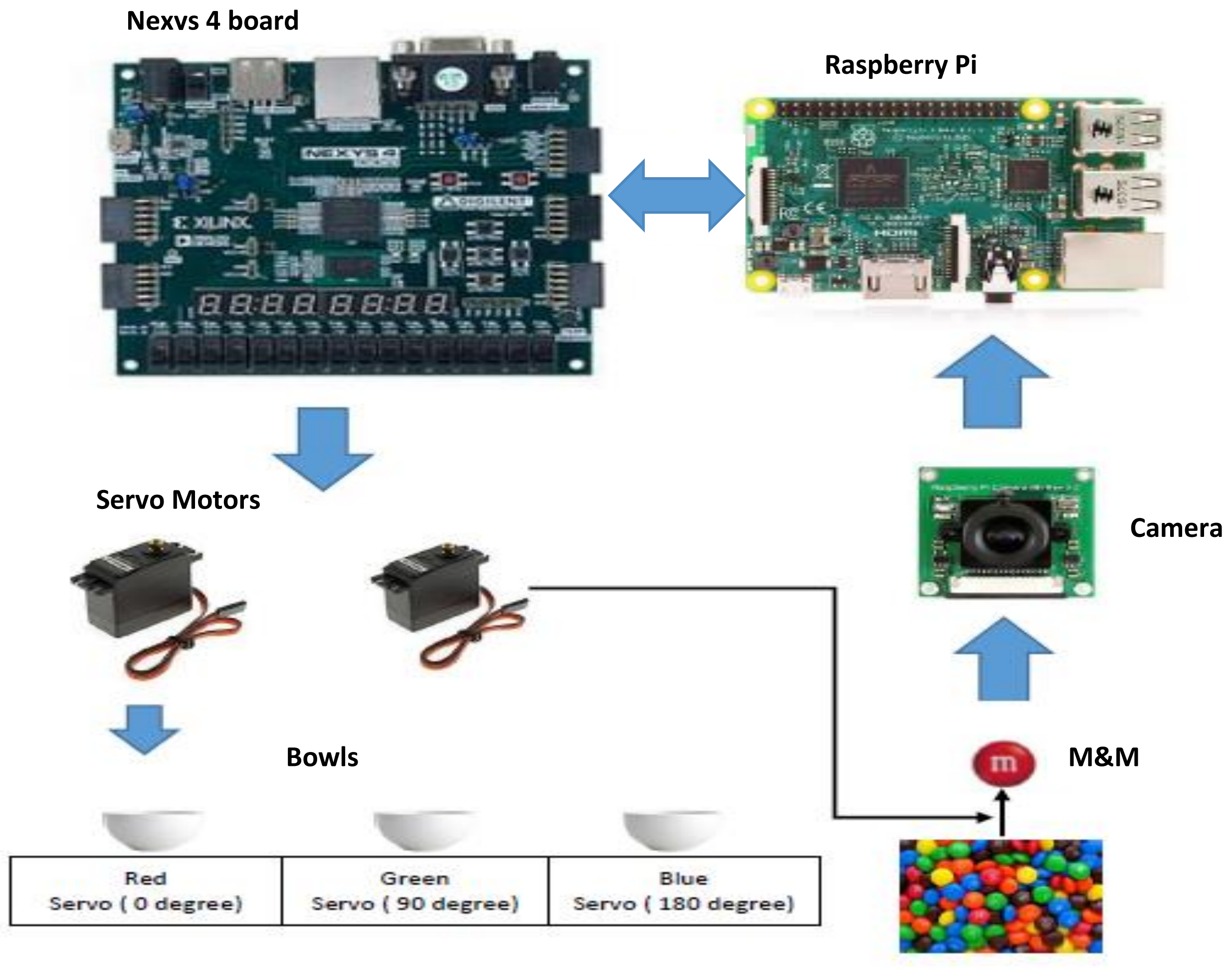

Fig. Block Diagram of the system

Raspberry Pi functionality:

- The Raspberry Pi was mainly used to integrate the camera with our system.
- As shown in the above block diagram it can be seen that the camera was connected to the raspberry Pi and was controlled to take pictures of the M&M.
- The Pi detected if there was any M&M kept on the board and marked where the M&M was.
- For this object detection, the Pi used a masking technique. We masked the frame and searched if any colors in the range of M&M color are available in the frame.
- If there was any similar color object detected, then it would check the area in which the color is spread out. Accordingly, it highlights the detected M&M in the frame capture.
- Once the object is detected, the Pi crops the image into a 28x28 Pixel image and converts it into a grayscale to be transferred for color detection.

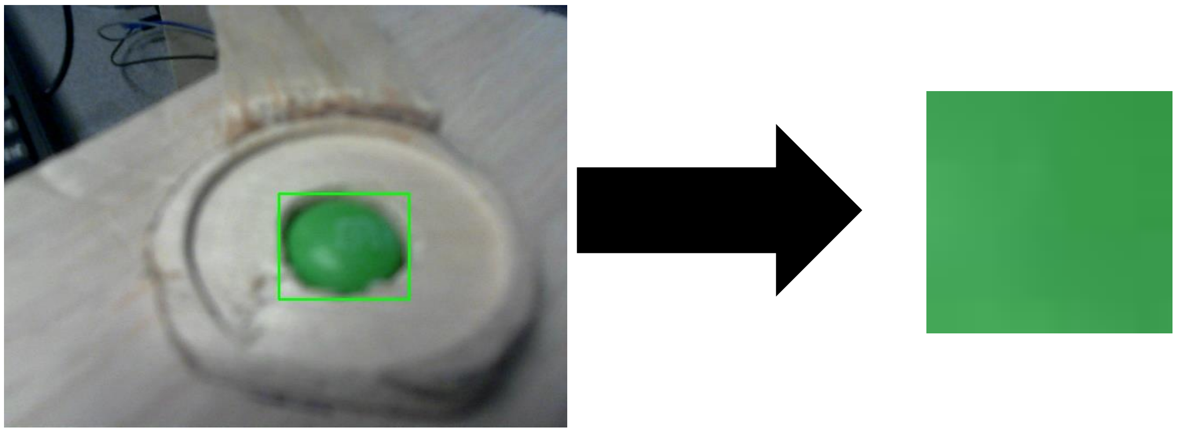

Fig. Detected object in the Raspberry Pi.

Nexys4 Functionality:

- The Nexys4 FPGA is the main controller of the system. It initiates the UART communication with Raspberry Pi by sending a signal.
- The Pi then does the processing as shown in previous section and sends the image back to FPGA.
- This image is then fed to the Neural Network (NN) built on FPGA and color is identified.
- According to the color detected by the NN, the servo motors are moved in order to drop the M&M.
- The detected color is displayed on the OLEDRGB display as shown in below diagram.

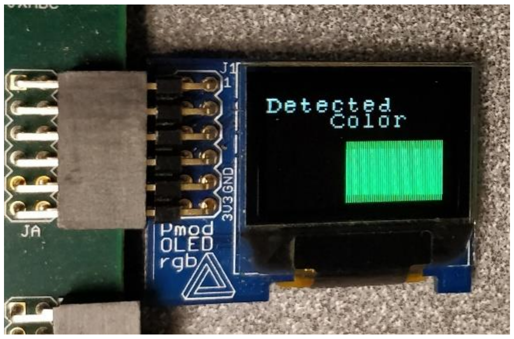

Fig. Detected Color.

Neural Network Implementation:

- The figure attached below shows the basic implementation of a neural network.
- The NN is divided into 3 different parts – Input layer, Hidden layer and Output layer.
- In this project, we gave the 784 pixels of the cropped image (as explained earlier) to the input of the neural network.
- Due to this, we have 784 artificial neural nodes aka neurons in the first layer.
- Then during the training phase of this NN we found that if we have 100 nodes in the hidden layer then we get a really good accuracy. So we chose 100 nodes in the hidden layer.
- As we designed this system only for three colors- Red, Green and Blue, we have three nodes in the output layer.

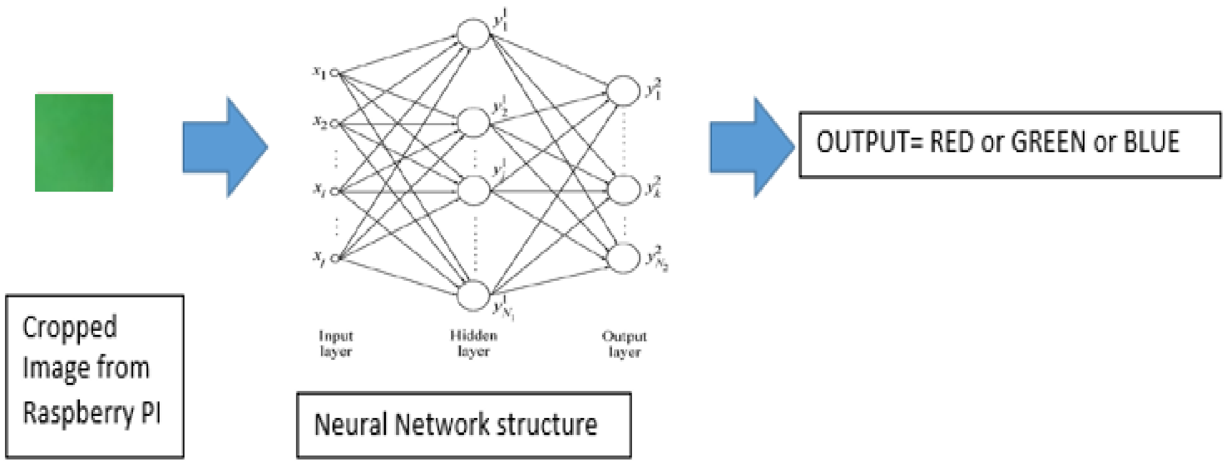

Fig. The Neural Network Overview

- The topology of neural network used in this system was a Dense Feed Forward Topology.
- This means that there was no feed-back loop between the different layers.
- Dense network means that every node in the current layer is connected to every node in the next layer as is shown in the above figure.
- As this is a feed forward topology, the input is fed to the input layer and output is checked at the output layer.
- Based on the different probabilities observed at the output layer, back propagation is done and the weights of each node are adjusted.
- The basic diagram of a single neuron is shown below.

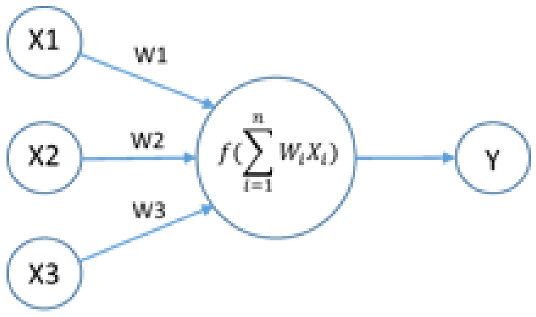

Fig. Basic structure of a single neuron

- The X_i shown in the above figure is the input to that particular node and the W_i are the weights of that particular input.
- Weights are basically a proportional constant which we adjust to determine how much a particular input affects the output. We do the adjusting during the training phase of the NN.
- The product of weights and their corresponding inputs is summed up together and fed to a function called as activation function.

- The neurons calculates output as per following mathematical calculations:

$$Y = f\left(\sum(Xi * Wi)\right)$$

- The function varies depending upon the different applications as well as depending upon different layers.
- Since our application was for classification purpose, we used a ReLu activation function for the input layer and the Softmax activation function for the output layer.
- The ReLu activation function is defined as below:

$$f(x) = \max(0, x)$$

- This means that the output will be either zero or positive. It would make output linear in nature and thus make the classification becomes easier. That is why this function is called as Rectified Linear unit (ReLu).
- In our application, this function was used in the Input and hidden layer.
- The Softmax function makes sure that all the outputs of the nodes in that particular layer would amount to 1. Thus making the outputs of all the nodes into probability of the classification.
- This is why this function was used in the output layer. As we would like to know the probability of the output prediction.
- The output color is decided by the node with highest probability.

Results:

- We designed and trained the NN on python using multiple libraries like Keras, SKLearn and compared the efficiency of the classifier generated by the respective libraries.
- We also considered the implement ability of the generated weight matrices and chose the best possible NN.
- We also tried different optimizers like Adam, Adagrad, SGD and RMSProp for optimizing the output in least size of the databases.
- Also we tested the NN for different lighting conditions of the data.
- We were getting accuracy around 73.33% with SKLearn whereas we got better accuracy with Keras.
- At the end, we chose the Keras library for implementing and training the Neural Network.
- We trained the neural network for 1000 epochs on multiple datasets of images of M&M's clicked from our hardware.
- We achieved **86.66% accuracy** with this setup.
- We then built the trained neural network in C as described in mathematical form in previous section.
- The layer implementation was done by 2D array multiplication.
- We successfully implemented all the goals that we set to achieve.
- The implemented hardware setup is shown below:

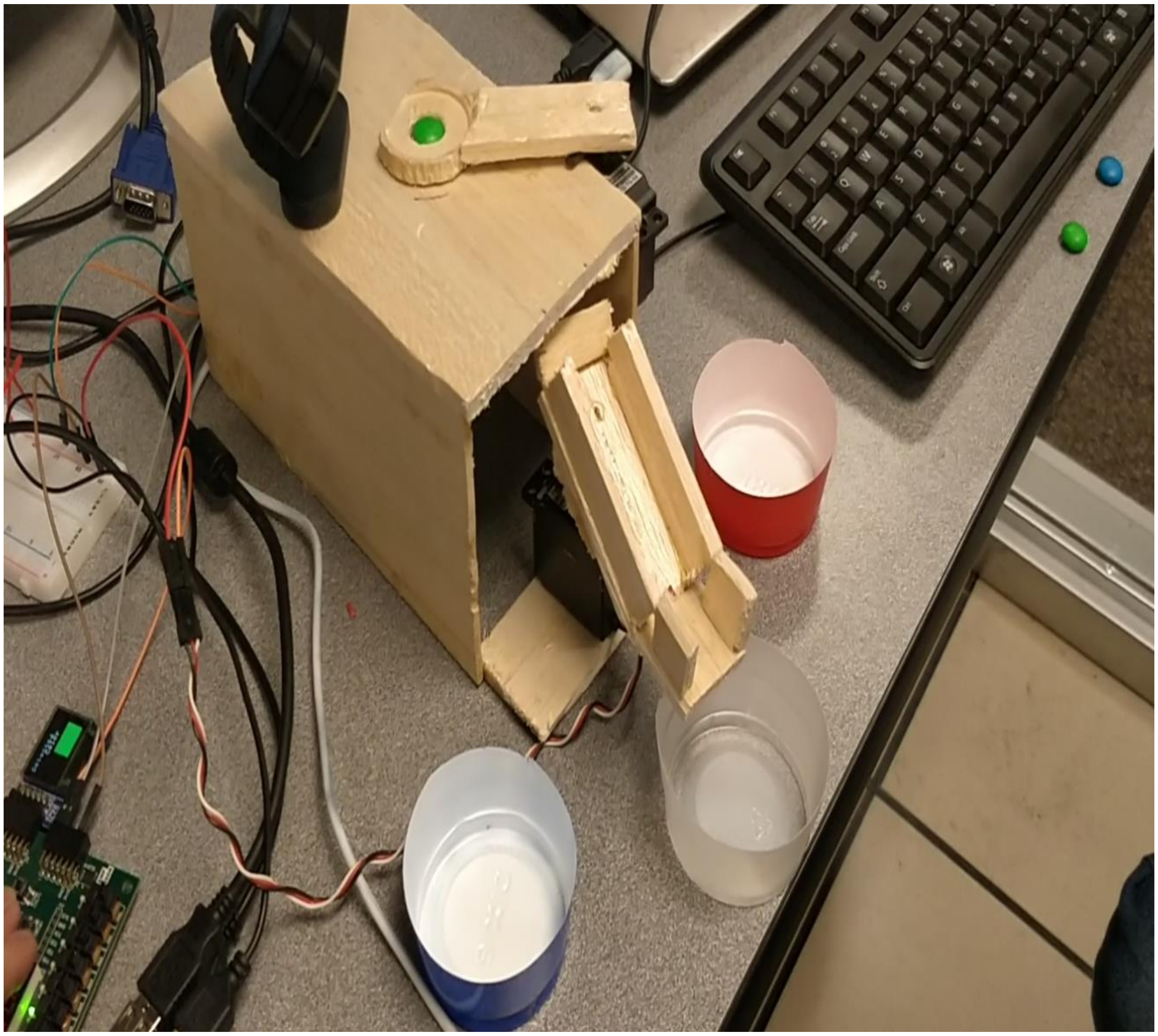
Fig. Setup implemented.

- The working video of the setup is attached in the repository.

Challenges faced:

- Training the neural network was a big challenge in our project.
    - Training is very sensitive to the lighting conditions of the data.
    - Very sensitive to the noise present in the data images.
- Object Detection in Raspberry Pi
    - Pi was detecting different objects with similar colors in the frame.
- Nexys4 implementation

    - We faced some challenge to store the weight data in Nexys4.
    - There are 78700 fixed point elements in the weight matrix amounting to 157400 fixed point multiplications and additions.

Stretch Goals:

- Add the training part of the project on FPGA.
- Add provision for more than 3 colors using dynamic programming.
- 3D print the case of the machine and make industrial level prototype.

References:

- Wikipedia.org
- Nexys4 Manual
- Xilinx Forum
- Keras library documentation
- OpenCV documentation
- Raspberry Pi Documentation
- Raspberry Pi forum
- TS-15 Servo Motor documentation
- Neural Network videos on youtube