Emika Hammond
Shamir Legaspi

Draw that Shape

"Draw that Shape" is an interactive learning game targeted toward children learning their shapes. The system will display an indicator for one of a select few shapes and give the participant a set amount of time to draw it. It will have a camera pointed at the drawing area and use a trained model to classify the drawing into one of the selected shapes and see if it matches the original shape presented. Feedback on whether the indicated shape and drawn shape match is given to the user.

Description of project

The game is played using a BeagleBone Black connected to a USB camera, 3 blue LEDs, 1 red LED, 1 green LED, and 1 push push button. The game starts when the player pushes the button. Then the shape to be drawn is indicated in the blue LEDs. One blue LED means the shape to draw is a circle, two means triangle, and three means rectangle. Now, the player has 10 seconds to draw the shape. Once the 10 seconds pass, the camera captures a frame of the drawn shape, and if the green LED turns on, it indicates that the player drew the shape correctly, and if the red LED is on, the player has drawn the wrong shape. If the player wishes to continue playing, they can press the button again to play another round.

Description of Components

Classifier Training

1. Take photos of the desired object to be detected in various conditions of lighting, angles, and board state. Ideally there should be hundreds to thousands. Organize the files into directories for positive and negative images.
2. Create a text file that contains the path to all of the negative or background images.
3. Use opencv_annotation.exe to create a file with all positive images and annotations of how many objects there are in the image and what the bounding boxes around the objects are.
4. Create a vector sample file from the annotated positive image text file using opencv_createsamples.exe
5. Train and retrain the model giving a directory and the background images files, positive vector, and tuning parameters for the number of positive and negative samples to train on, the number of stages to train, the minimum hit rate, and the maximum false alarm rate using opencv_traincascade.exe

Kernel module shape.c

1. Detects button press signals that signal the start of a game round.
2. Generates a random shape that the user must draw on button press. The kernel can set 3 blue LEDs depending on the shape generated. (1) blue LED indicates a circle, (2) triangle, (3) rectangle.
3. Sets a 10 second timer at the start of the round on button press.
4. The kernel module sets a flag `READY_FOR_USER` when the timer expires. It indicates the user program can proceed to capture and analyze the camera frame.

5. When the `READY_FOR_USER` flag is set, the data sent to the user when the device file is read from notifies the user program that it can proceed to capture and analyze the camera frame.
6. The kernel receives classification results from the user program when the device file is written to and unsets the `READY_FOR_USER` flag. This ensures that the camera frame is only captured and analyzed once per button press.
7. If the user program predicts the correct shape, the kernel sets the GPIO pins to high or low.
    a. Green LED is activated when the drawn shape is correct.
    b. Red LED is activated when the drawn shape is incorrect.
8. If the user presses the button again after the round, the module clears the prediction results, resets the timer, and generates a new shape to be drawn.

User program shape.cpp

1. Loads pre-trained cascade classifiers for detecting circles, triangles, and rectangles respectively
2. Constantly reads from the character device in a while loop, waiting for an indication that the kernel is ready to receive an output from the user program. If the kernel is not ready yet, the program goes back to the top of the loop.
3. Captures a camera frame and inputs it into the classifiers to determine the most prevalent shape. If there are ties for this or the incorrect shape is determined to be the most prevalent, the shape the player drew is considered incorrect.
4. Writes the predicted shape to the character device based on the classification of the cascade classifiers.
5. After writing to the character device, the user program waits until the kernel module is ready for user input again.

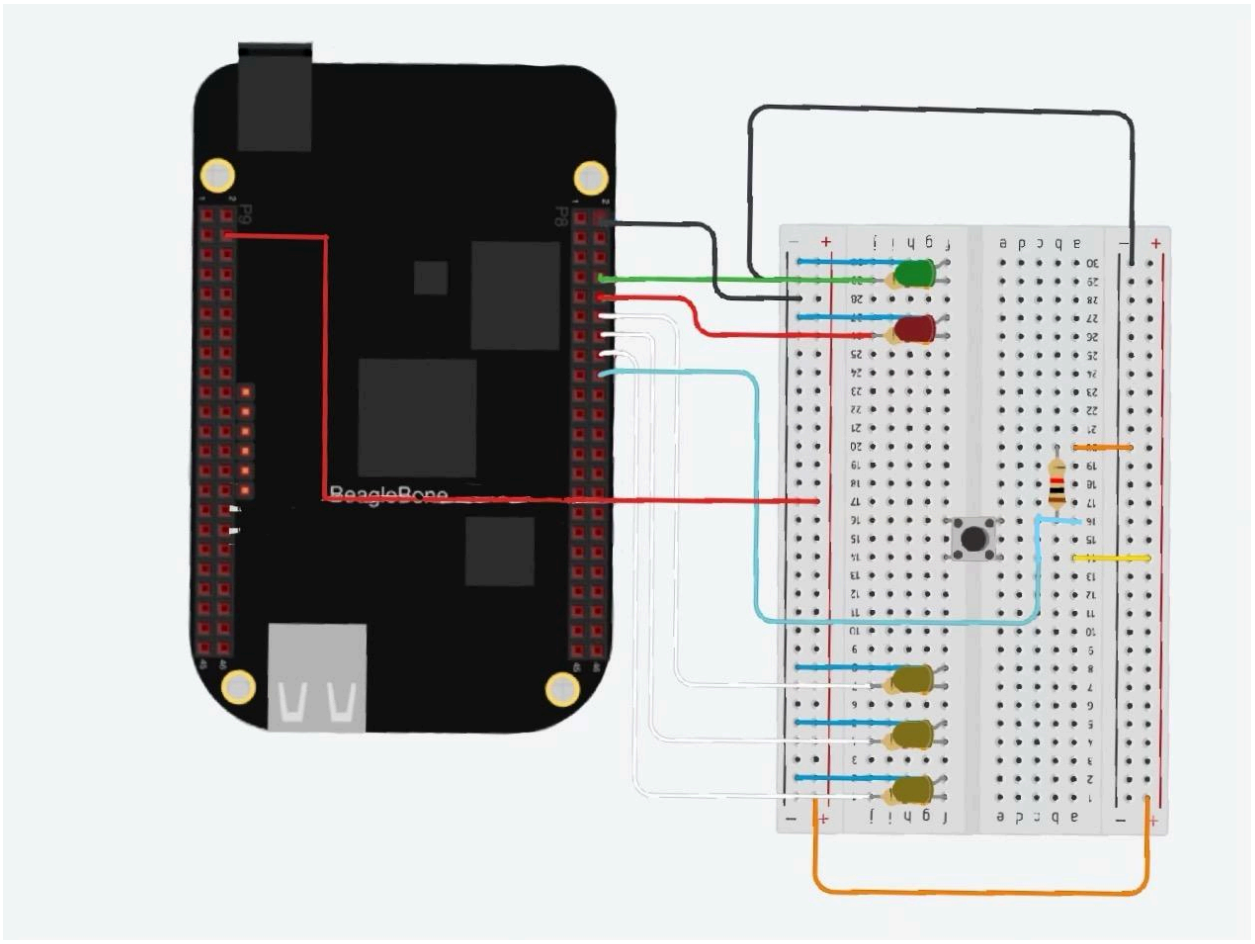
Draw that Shape Startup Configuration for BeagleBone Black (BBB)

1. Created a shell script called `run_shape.sh` to handle setup tasks for Draw that shape. These tasks include creating a character device, loading the game's kernel module, and starting the user program in the background. Initialization information for systemd is also included within the script placed in `usr/local/bin`.
2. Define a systemd service called `shape.service` to start the game on BBB startup. The definition was placed in `/etc/systemd/system`.
3. Reloaded systemd to recognize new service. This was accomplished by running the command `sudo systemctl daemon-reload`.
4. The shape service was enabled using the command `sudo systemctl enable shape.service`. This ensures the service starts automatically upon system startup.
5. Rebooted the BBB to apply and check these changes. Upon system startup Draw that Shape was initialized automatically.

Aspects of Project not Implemented from Original Project Proposal
One aspect of the original project proposal that was not implemented was outputting game information to an LCD screen. This is because it was difficult to work with the LCD screen and the serial connection. Upon inserting the LCD screen into the BeagleBone Black, it was evident that the given serial connection interface could not be used at the same time as the screen cape. Given that the BeagleBone Black and serial connection are property of the university, the implementation was changed to display the game's output to LEDs rather than modifying the connection interface. A better classification/image detection model could also be used. Under certain conditions, the classifier has low accuracy.

Functionalities Added in Addition to Original Project Proposal
One aspect of the project that was not in the original proposal was using Debian as the operating system and running the Shape game on startup of the BeagleBone Black. It is more intuitive as a game that works right out of the box, and is the point of an embedded system, it is ready to use.

Software Architecture / Schematic



What We Learned from EC535 Final Project
In this project, we learned how to use opencv in c++ and through the command line. From this, we learned what specific commands and functions to call to train and use a cascade classifier.

Moreover, we learned how to collect data and change training parameters based on the data available. Changing the number of stages for training, the target hit rate, and the target false alarm rate all impacted the output of the model. We also strengthened our problem solving skills and implemented the many components throughout the class into a comprehensive project. Naturally, we used a kernel module for real time events, a user program for more computationally intensive, less time sensitive events, and a character device to communicate between the two. When we ran into an issue with using an LCD screen, instead of giving up on that functionality, we chose another method of implementation. Finally, we ran into an issue using the startup service we learned in the lab, but still got the desired functionality by looking through documentation.

References

Flashing BBB with debian
https://www.beagleboard.org/distros/am335x-11-7-2023-09-02-4gb-microsd-iot
Training a Cascade Classifier through OpenCV
https://www.youtube.com/watch?v=XrCAvs9AePM&ab_channel=LearnCodeByGamin
Fixing linker command error upon compilation of program using OpenCV
https://forum.opencv.org/t/linker-command-fail-in-macos-12-5/11962
Example of cascade classifier detection in C++
https://docs.opencv.org/4.2.0/db/d28/tutorial_cascade_classifier.html
Generate random number in kernel module
https://stackoverflow.com/questions/12961299/generate-random-number-in-kernel-module
Init.d script in debian example
https://gist.github.com/drmalex07/298ab26c06ecf401f66c
Systemd service failing due to `Default-Start contains no runlevels` error
https://askubuntu.com/questions/909523/default-start-contains-no-runlevels-aborting