Smart Alarm

Cody Bourgeois and Jeremy Desforges

Project Description:

Our project goal is to develop a smart alarm system that includes multiple aspects of waking the user up and preparing them for their day. Before going to bed, the user will enter their desired time to be woken up to the alarm. Once the alarm is activated, the car will evade the user around their home while also enabling different smart home features depending on the environment. If the user is chasing the car and they enter the kitchen, the car may tell the toaster and coffee maker to start making breakfast. When the user has finally caught the alarm car, they will have the lights on, blinds open, and a fresh breakfast waiting for them to start their day.

Project Materials:

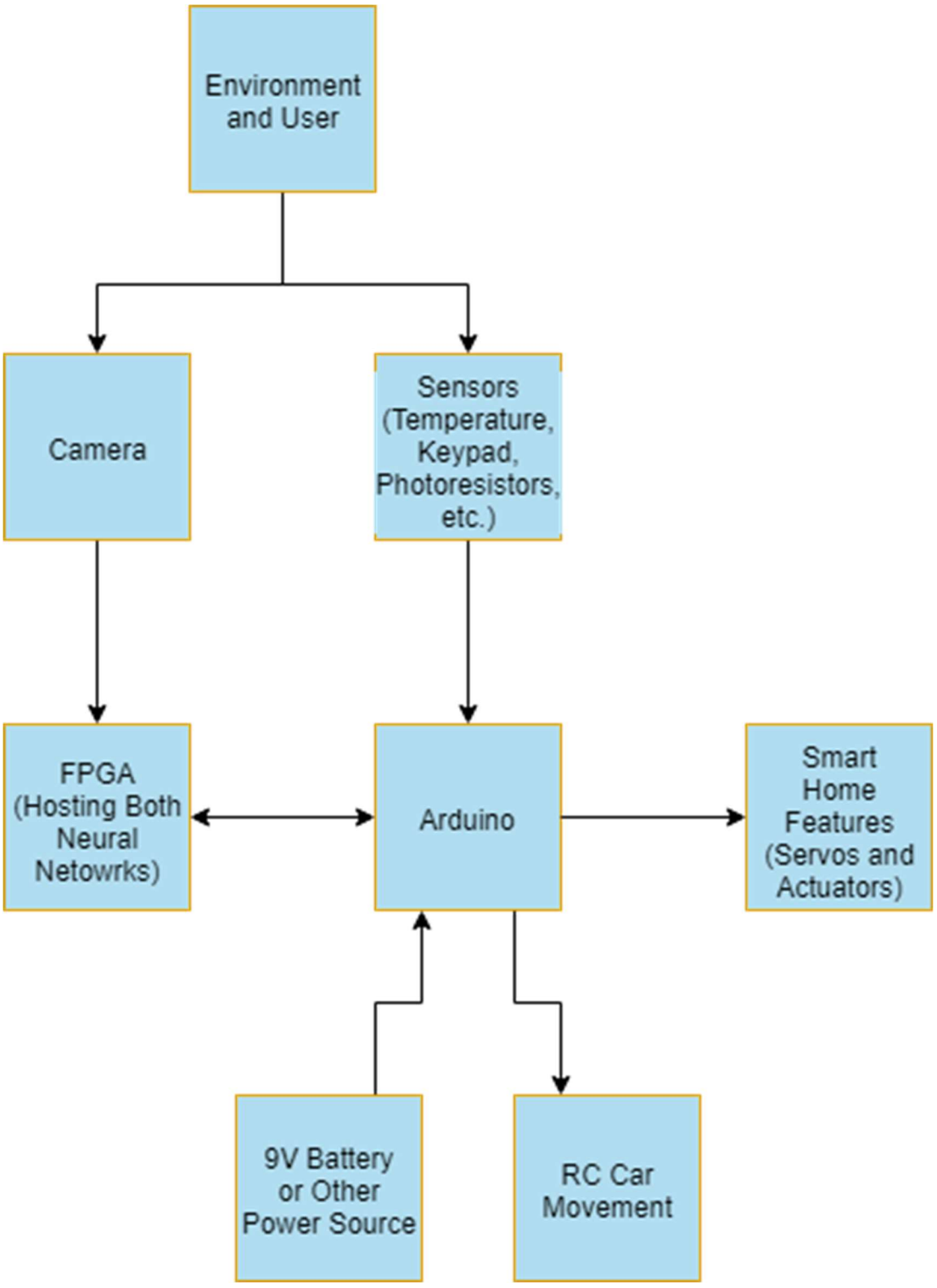
Hardware:

- Arduino Uno
- Arduino MKR Vidor 4000 (FPGA)
- 9V Battery or other power source
- Speaker
- Keypad
- LCD Screen
- RC Car or components to build one (Wheels, motors, body, etc.)
- Camera
- Photoresistors
- Temperature Sensor
- Servo Motors - For Shower, Toaster, Keurig, Blinds, Thermostat, Lights
- Toaster, and Keurig or other Coffee Maker

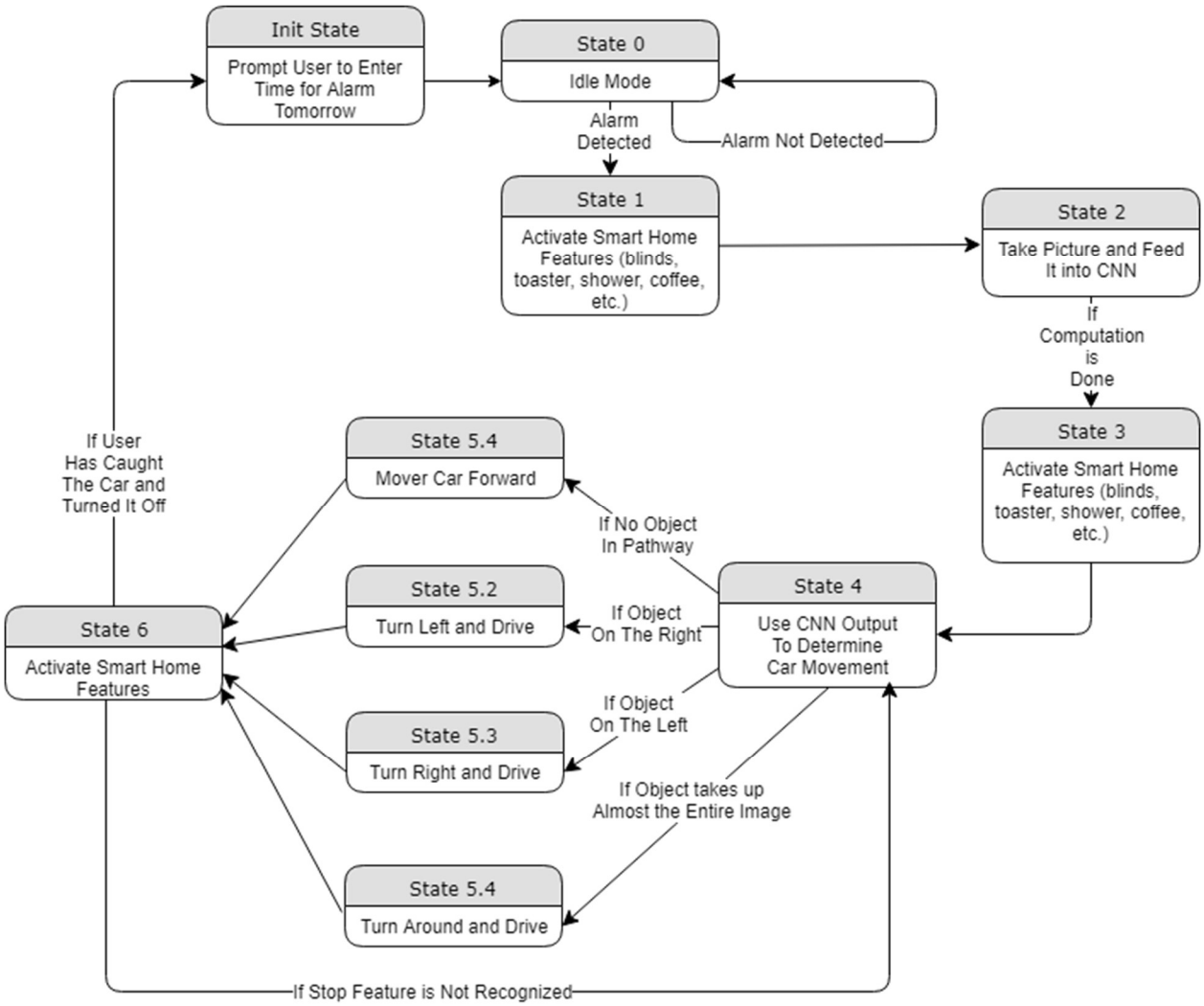
Software:

- Python - TensorFlow and Keras
- ModelSim - Verilog
- Quartus Prime - Compiling Verilog Code
- Arduino IDE - Programming microcontroller and FPGA

Project Block Diagram:



Project State Diagram:



Machine Learning Model:

Our project will include two machine learning models, which will be accelerated using the Arduino FPGA. The first model will input sensor data from the temperature sensors and light sensors outside, time until the user must leave for work, and the user's location relative to the kitchen or bathroom. The outputs of this basic feedforward neural network will be used to make decisions for the thermostat setting, dimmer light levels, turning the shower on, making toast or coffee, and whether or not to turn the user's car on for them. The goal is to get the neural network to learn the connections to help the user get ready as fast as possible if they are late, save energy by setting the thermostat and lights appropriately, and if they have enough time make the user a light breakfast. I have already created a simulated dataset for 1000 examples and trained a very small neural network to have a mean squared error of less than .003 when predicting all of the outputs for 250 examples.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_5 (Dense)              (None, 8)                 40
_________________________________________________________________
dropout_4 (Dropout)          (None, 8)                 0
_________________________________________________________________
dense_6 (Dense)              (None, 16)                144
_________________________________________________________________
dropout_5 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_7 (Dense)              (None, 7)                 119
=================================================================
Total params: 303
Trainable params: 303
Non-trainable params: 0
```

Current Smart Features Network Model Parameters

The other neural network our project will utilize is a convolutional neural network for object avoidance. Because the alarm will be attached to an RC car that avoids the user, we will want the car to autonomously avoid all possible objects while it evades the user. With a camera attached to the front of the car, the camera will take images and feed them into the convolutional neural network hosted on the FPGA. The neural network will only classify images into two classes, Object or No Object. The images will be broken into 7 vertical slices and fed individually to the network to detect if an object is within the frame or not. The network will output a confidence score from 0-1 representing how confident the network is that there is an object in that slice, with 1 being 100% and 0 being 0% confident. We chose to use 7 slices to represent our field of view because this output can fit within one byte of data, has an odd number so there is a clear center of view, it will reduce the size of the actual neural network, and the size of the slices will contain enough information for the network to learn the required information. For the purposes of demonstration, we will only be training our convolutional neural network on images taken in the classroom that we will be demonstrating the final project.

Project Phases:

Phase I:

Create Arduino program to interface with the FPGA to accurately and quickly process the simulated input data as well as pictures of the user's environment to then simulate activating the appropriate actuators.

Phase II:

Connect Arduino to a real RC car to control its movements and improve the object avoidance algorithm using the convolutional neural network.

Phase III:

Enable extraneous smart home features with real sensors and actuators including the blinds, lights, thermostat, coffee pot, etc. Collect real data for improving neural network and send it to a cloud server or other data storage device.

Phase IV:

Improve neural networks using data collected from real use of the system and add more smart home features.

Roles:

Jeremy:

- Create fully trained neural networks for use in the full system
- Convert neural networks from python to verilog for implementation on the FPGA

Cody:

- Create Arduino functions to control the RC car using the outputs from the convolutional neural network
- Create functions to simulate input data to the smart features network and use the network's output to simulate actuator responses

Both:

- Get camera to interface with the FPGA and accurately preprocess the images
- Overcome any challenges we might face during implementation of the project
- Continue brainstorming for new ideas that could be included in the project

Challenges:

One major challenge for our project is training the convolutional neural network on a small dataset. Because we are limiting ourselves to the environment of the classroom, we hope that the network will be able to accurately train on our dataset without overfitting too much to the few objects in the room.

Another challenge is actually converting the trained networks to function on the FPGA. Writing the Verilog code might be difficult for the convolutional layers depending on how limited our resources are on the FPGA. If we can ensure that our network remains a very small size without losing too much accuracy, we should be able to overcome this challenge.