Occupancy Monitor

Open Option Project - Spring 2018

Mitch Avis, Lane Barnes, Shane Beuning

Summary:

This project is a continuation from Fall 2018. Our goal last semester was to design and build a device that could be mounted at the top of a doorway and accurately keep a count of the number of people in a room. We were able to successfully build a prototype, write the necessary software, and demo it at the end of the semester above one doorway to the BC-Infill, with a laptop displaying the current tally. This semester we changed the board type that we are using to one that has wireless capabilities. We built a second device that wirelessly communicates with the other board in order to synchronize the tally with multiple doors. We have also made changes to our algorithm to count two people going in or out at the same time. Finally, we have added a seven-segment display and battery pack so that the devices can operate and display the current tally when not connected to a laptop.

The Idea:

Continuing our project from last semester, our original idea was to build a compact device using some flavor of Arduino and a thermal/infrared (IR) camera module, that could be mounted at the top of any doorway. By taking only the thermal data from the camera, we can binarize it (shown in Figure 1) to simplify our algorithm. Binarization is useful to differentiate a potential person from the ground they stand over. When we analyze each frame, we only look at pixels that are above a certain threshold. This way, we can process the frame efficiently and without too much noise. The software can then determine whether a person moving in the camera's field of view was entering or leaving the room and increment or decrement the internal counter appropriately. We also attached a four-digit seven-segment LED display to the device so that it could continuously show the current tally. We also attached a battery pack so that it will run on its own above a doorway. The new devices are shown in Figures 2 and 3.

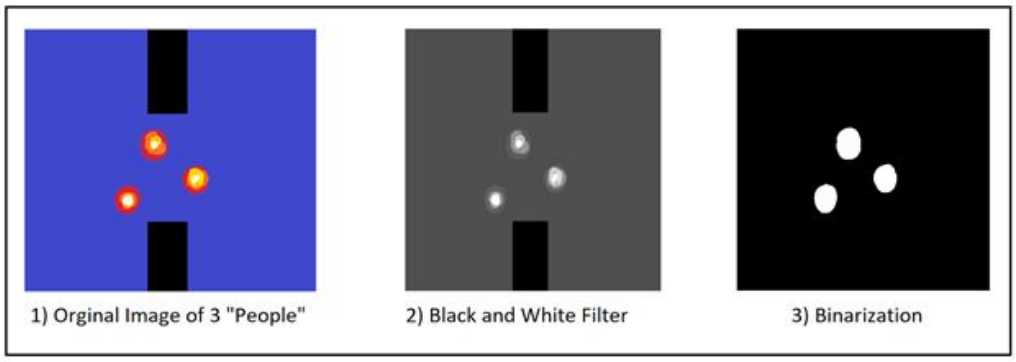

Figure 1: Counting people using binarization

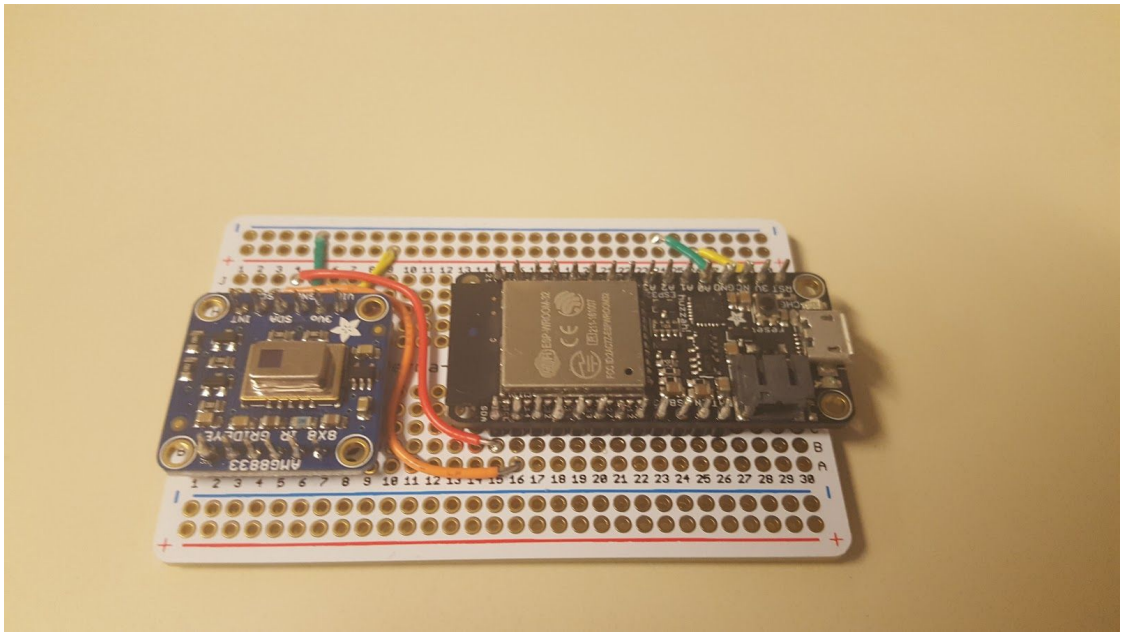

Figure 2: New device, with an IR camera and Adafruit ESP32 feather board connected via a soldered protoboard

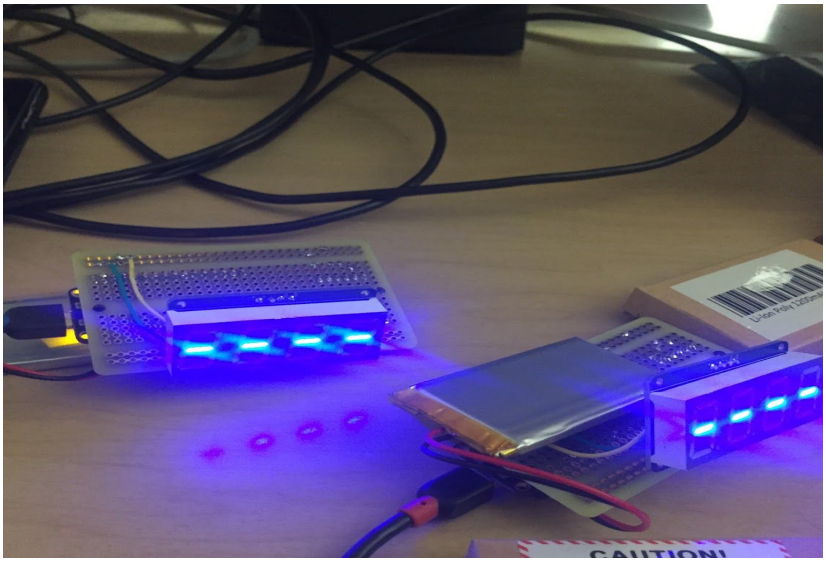

Figure 3: Two devices, with battery packs and seven-segment displays attached

Tracking Code:

Last semester's code worked by looking at all the pixels above the threshold temperature and determining the position of each pixel on the x-axis. We kept track of the total number of valid pixels in each column and calculated the total average position, X. This approach was very simple and straightforward for a single person, because the person's average X value is the same as the entire grid's average X value. This is a large problem for two people however, as each individual's average X value is not equivalent to the grid's average X value. We needed a new algorithm to fix this problem.

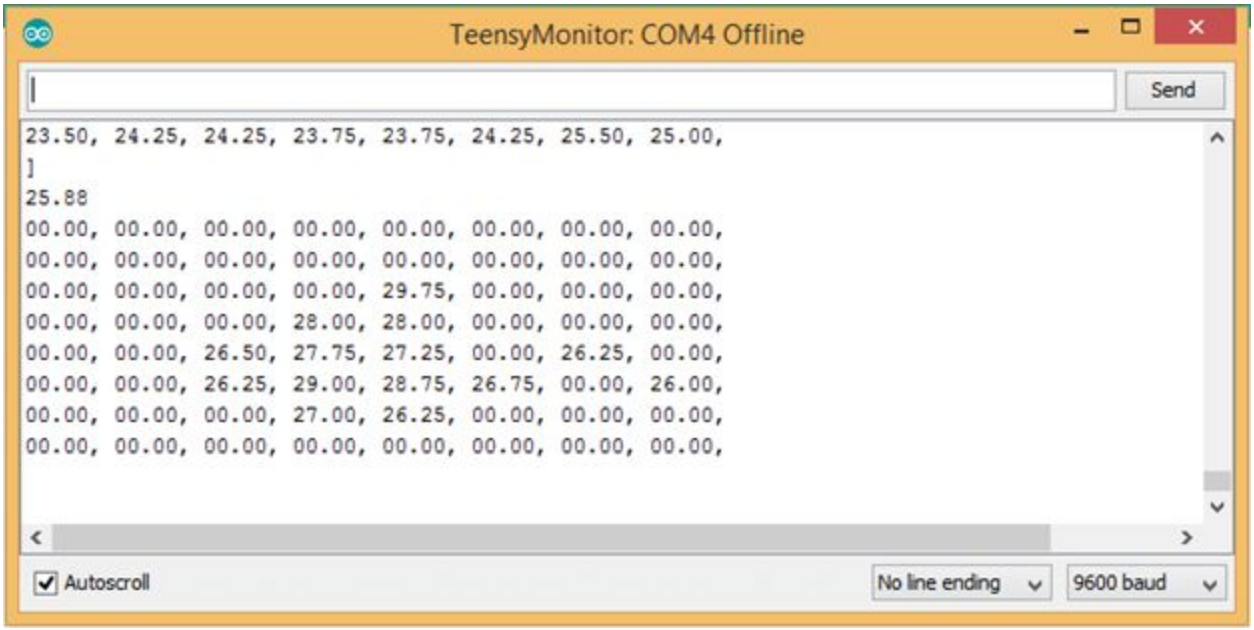

Figure 4: One frame captured from the camera of Mitch walking through the door

Our new approach is to look for local maximums instead of looking for averages. We cannot easily take the average X value of one out of two people because we have not differentiated the two people from each other. When two people are walking one after another through the door, each of them has their own local maximum of heat, which we can look at to determine if these are actually two people or not. If they are too close to each other, they will count as one person. We cannot say that they are two different people unless we get a better resolution camera (our current resolution is 8x8 pixels, as shown in Figure 4). Furthermore, if the local maximum is a part of a heat-generating body that is too small to be a human (for example, a cup of coffee), we do not count it.

In each individual frame, we look for these local maximums, and if they are big enough, they will count as a person. We will create a "Person" object and assign it to that local maximum. On the next frame, this local maximum will have moved slightly. We first look at this frame and find the local maximums at their new positions. We can then take the old position of Person1, find the nearest local maximum, and assign the new local maximum to be the new position of Person1. In this way, we can then determine if Person1 is indeed a person, and which direction through the door they are traveling. When they exit the frame and no longer count as a person, we can safely delete Person1, increment/decrement the counter, and continue to track a Person2 until they do the same.

The threshold temperature algorithm has changed this semester as well. Last semester, we determined the threshold temperature to be the total average temperature of all pixels, plus an arbitrary variable, called the ThresholdOffset. Taking the average of every pixel was our attempt to find out what the "ambient" temperature was. This worked well for one person, because when one person was in the camera's field of view, the average temperature remained close to the temperature of the floor, since each person does not take up much space in the frame. When there were two or more people, however, the average temperature became closer to that of body heat, making the threshold temperature too high to be useful. Again, we needed to rethink this code to track more than one person.

The new threshold calculation now looks at each individual pixel, instead of all the pixels. In addition to the array that keeps track of pixel currently viewed by the camera, we have a separate array that keeps track of the minimum of each pixel. The minimum of each pixel is typically the temperature of the floor. The threshold will now equal the minimum of the pixel plus the ThresholdOffset. This makes the binarization process much more efficient, as we are now able to have a stable and predictable threshold.

Another problem we faced is people carrying cold beverages past the doorway. When the camera saw the cold beverage, it would lower the minimum temp lower than the temp of the floor. This meant that the threshold would be lower than normal, and the tracking code would detect objects that were not actually hot enough to be people. To fix this, we slowly increment the minimum temperature over time, so that, if a cold beverage does pass the camera, the minimum temperature will slowly rise back to the temperature of the floor. This also helps with situations where the temperature of the floor increases as the day goes on.

Wireless Code:

To incorporate the new boards' wireless capabilities, we used a pre-existing Arduino library and modified it heavily to suit our needs. The library originally allowed us to set one device as a "Slave," and another device as a "Master." The Slave device would establish a new wireless network and attempt to receive any bytes of information that might get sent to it, while the Master device would attempt to connect to any existing Slaves and send information. Due to the nature of this connection (one device only sends information, while the other only receives information), we needed to devise a method of allowing both devices to send and receive. To solve this issue, we named each device APn, for Access Point n, where n was the individual device's assigned ID number. We made it so each access point creates a network (named Occupancy_Monitor_n, where n is that device's ID number), then scans for any other existing Occupancy_Monitor networks in order to establish a connection. When two devices establish a connection with each other, they are each then able to send and receive information to one another. Once this connection algorithm was implemented and working, it was a fairly straightforward task to have them share their current count of people in the room, receive another device's count, and combine the number to be displayed and incremented or decremented in the future.

Results:

For testing purposes, we placed the devices over the doorways of lab rooms C105 and C107 and get them effectively and accurately communicating with one another. However, when we had them above the doors in the BC-infill for our demonstration, we ran into some issues associated with one of the devices. Initially it was communicating, counting, and displaying correctly, but after a while, we noticed that it had stopped counting movement through the door. The seven-segment display was still matching our working board, so communication and synchronization were still taking place, but the device's own tracking was not working. After the event, we found that the board had switched into debug mode, which caused the camera to malfunction. Even with this unfortunate error occurring during the showcase, we still had working communication and counting on two boards. For potential future improvements, we also made it easy to scale up the number of devices working together, using the same code.

Future Possibilities:

Due to the initial excitement over the seemingly limitless possibilities this project presented, we purchased enough supplies this semester to build up to six devices. Our plan was to have the ability to connect all six devices in a mesh network, and configure them in such a way as to not only monitor the number of occupants in a given room, but also be able to track the flow of traffic around a building (with interconnected rooms). Other ideas we had included: designing and printing sleek and functional cases for the devices using 3D printers; designing a mobile app with the ability to connect to and configure each individual device remotely, as well as to read real-time monitoring data; and finally, upgrading the hardware in such a way to make the individual components all part of a custom-built integrated circuit board (ICB) that could be manufactured and marketed (instead of using open-source yet proprietary Arduino products). Some of these ideas are still pretty far from being realized, but we are hoping to either continue working on it or pass it along to another team that might be excited about it as we were.

Concepts Applied from Other Courses:

To get the software side of things going after we had our prototypes built, we used a fully functional library that came with the thermal camera and that was fully compatible with the Arduino. Using this library, we were able to learn several new concepts of C++ programming, which helped us to expand the program in order to develop the algorithm we needed to track movement and update the counter. Furthermore, we incorporated several different techniques that we have learned throughout our schooling in order to help us efficiently build our prototype. We used our knowledge of algorithms and data structures (from various Computer Science courses) to help us develop the program and its efficient method of displaying each frame's values. We initially used concepts from linear algebra to help us determine the correct values of the matrix to be used for tracking heat objects, and while we did end up changing our approach to this challenge, our first method of doing this ended up being a solid launching point. We also used concepts from ECE classes and labs to help us construct the prototypes on a breadboard, wiring the components together, and soldering the finished connections.

Conclusion:

For the two semesters that we have been working on this project, we have learned applicable skills that can be carried on to the real world. While working on this project, we often ran into issues regarding a difference of ideas on how to implement the base level of our code and design. This was a large time sink that we could have dealt with in a much smoother fashion, but by the end, we learned how to communicate our ideas more effectively and decide on whose ideas are better by a democratic, pros and cons process. While working on this, we also should have spent more time in the testing phase, instead of worrying about less consequential issues, like getting the display synchronization 100% accurate. When we first tested it, we saw that it worked and assumed it would work equally well during the demonstration. However, we did not run it for different cases, like leaving it up and running for an extended period of time, testing with heavy traffic coming in and out, or testing with different ambient temperatures. If we had taken more time to test these and various other cases, we could have been better prepared. With this in mind, however, we did reach our goals of having wirelessly connected devices with working cameras, rechargeable batteries, and LED displays, and we learned useful skills related to Arduino devices and programming.

Bibliography

Niklas, Felix. "Binarization." Image Processing - Binarization, 2017, felixniklas.com/imageprocessing/binarization.

Adafruit. "Adafruit/Adafruit_AMG88xx." GitHub, 29 Oct. 2017, github.com/adafruit/Adafruit_AMG88xx.

SensorsIot. "SensorsIot/ESP-Now-Tests." GitHub, Dec. 2017, github.com/SensorsIot/ESP-Now-Tests.