waitr

Team E6: Sophie Sacks, Dina Razek, Samantha Lavelle
18-500 Capstone Design, Spring 2022
Electrical and Computer Engineering Department
Carnegie Mellon University

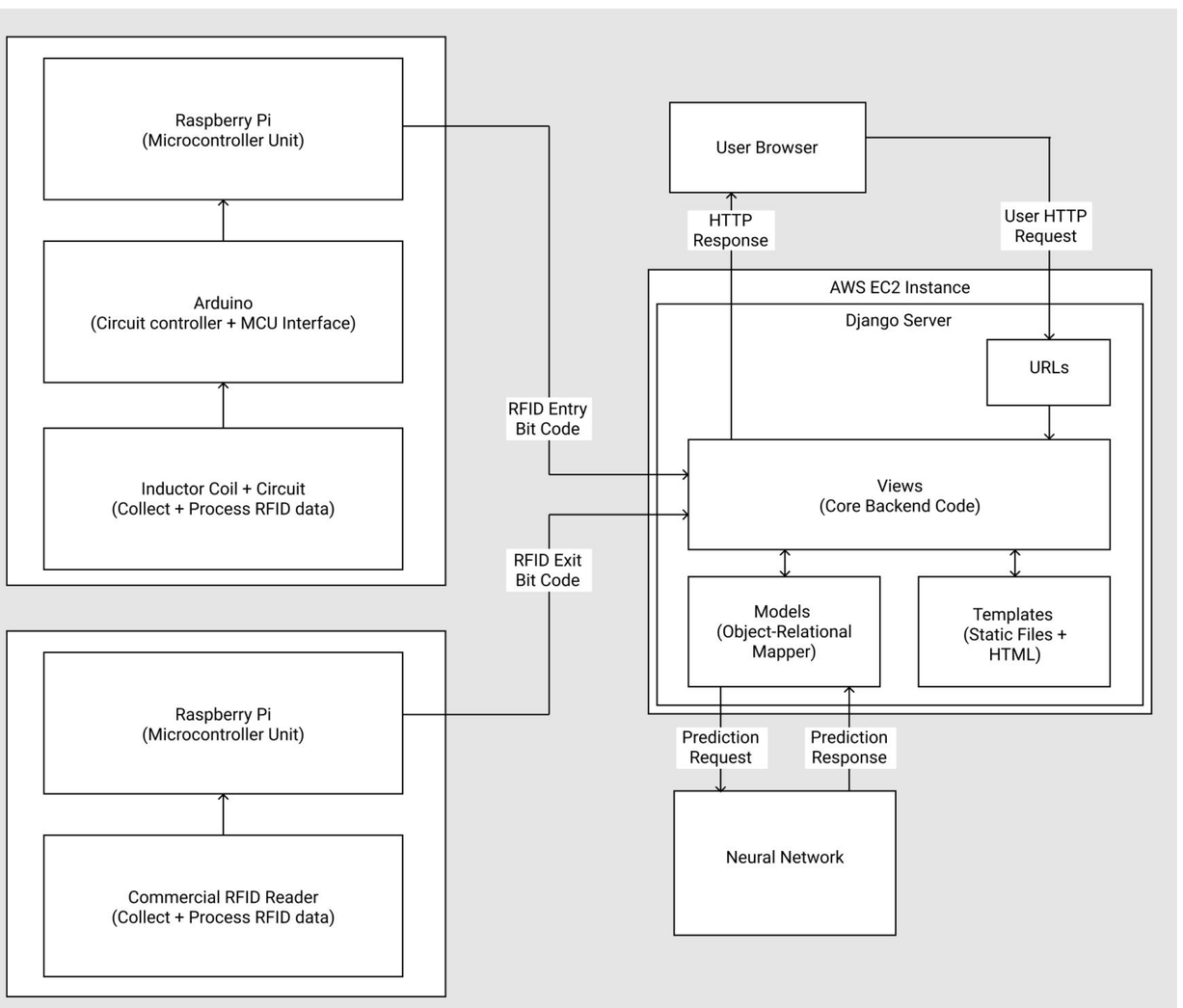
Check out our web application here! https://waitr.link

Product Pitch

Students on campus have a difficult time knowing how crowded restaurants are in real time. A combined hardware and software solution helps mitigate this issue. Our end product consists of two radio-frequency identification (RFID) scanners that track how long each patron is in line: each user scans in at the beginning of the line and scans out at the end of the line. This data is sent to our web application, which uses a machine learning (ML) neural network to predict wait times using this live data and display the current live and approximate wait times. Our system transmits data from the RFID reader to the web server in under 2 seconds, and our web application responds to the user in under 2 seconds, as well. The accuracy of our wait time predictions has a margin of error of 10%, or 2 minutes. Our entire database can keep track of up to 50 patrons.

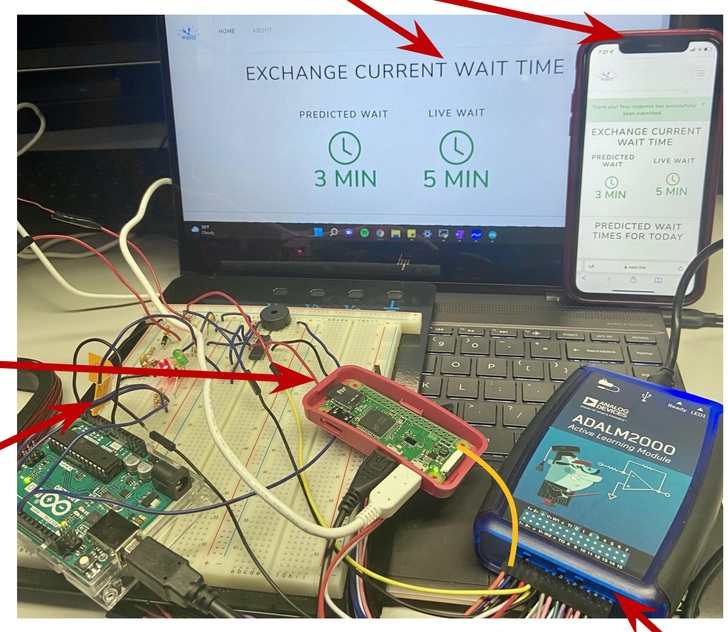
System Architecture

The figure below displays our entire system architecture, including the connection from the hardware (left) to the software frontend and backend (right). An Arduino is connected to the circuit to convert the ID bit code to digital and send to the Raspberry Pi Zero. The RPi used to send the scan time and the ID value as a POST request to the web application server. Our web application uses a Django Python-based backend framework deployed as an EC2 instance on AWS. The *views.py* file handles our backend logic, connecting our models, forms, and HTML templates. Our neural network uses trained and live data to predict wait times.



System Description

Our system consists of a custom circuit to collect the bit tag from a 125 kHz RFID tag, a Raspberry Pi to transfer the bit tag to our web application, and a machine learning model that factors the scanned in and out time into the current wait time calculation. The circuit collects the bit tag via a custom antenna that modulates at 125 kHz; a demodulator stage; and a filter stage that isolates the signal of interest. This analog signal is then input to an Arduino, which translates the signal to a digital bit code and sends it to an RPi, which makes a post request to our web application. If the ID number scanned is stored in the database, both the entry and exit times will be saved as a *WaitTime* object. Otherwise, the entry time and ID number will be saved as an *Entry* object. Once a week, the *WaitTime* data is provided to the neural network for training. This ML model takes in the day of the week, the hour, and the minute and predicts a wait time in minutes.



Web Application that displays wait time data

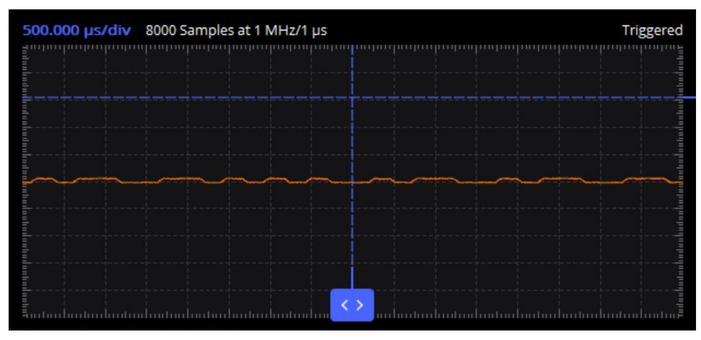
RPi to send bit code to web server

Circuit to gather + demodulate, filter RFID bit code

ADALM, which provides signal output + power to circuit

System Evaluation

Our circuit was evaluated via a number of component tests conducted both with lab equipment and an ADALM. The inductor coil resonance was tested by introducing a 125 kHz RFID tag and ensuring that the output at this stage was modulated when doing so. The filters were tested via a frequency sweep, in which we verified that the output begins to attenuate only after our cutoff frequency of 16 kHz. This confirms that desired signals are acquired, while any unwanted noise is filtered out. Finally, the output of the circuit was analyzed with an oscilloscope, and as an RFID tag was introduced, its bit tag was seen on the monitor. This confirms that our circuit successfully acquires, demodulates, and filters a signal from a 125 kHz RFID tag.



Output of the circuit in the presence of an RFID tag. Captured via oscilloscope.

To ensure we met the use case of quick transmission of data, we printed the time it took for the server to receive the data from the scan time, and based on 50 data points, we found that the average time is 0.96 seconds, which satisfies our design requirement of within 2 seconds. To test our web application response time, we printed the time it took for a page to reload based on updates to the data, and found that the average time from 50 data points is 1.84 seconds. To ensure we could keep track of all the data coming into the system, we tested to see if we could hold 50 separate data points and found no errors or loss of data while doing so.

Conclusions & Additional Information

http://course.ece.cmu.edu/~ece500/projects/s22-teame6/

Overall, we were able to complete a fully working product that resembles our original design. To further improve this project, the circuit's target frequency could be modified to extract data from real CMU ID cards, and this design could be translated to a PCB. This would allow the RFID scanners to be installed in an on-campus restaurant and used in real time for our intended application.

Electrical & Computer ENGINEERING
Carnegie Mellon