

OurAbode is a piece of software that is designed as a prototype for a system that allows users to interact with their home in a more comfortable and intuitive way. We feel that everyone should be able to live in comfort, and in this technology-enabled age, this has never been easier. With this system, users will be able to customize their homes as autonomously or hands on as they wish.

**Hardware**

The OurAbode system uses primarily Arduino and Arduino-compatable products to allow for a quickly assembled prototype. It utilizes the Adafruit PN532, an arduino shield used for RFID and NFC reading/writing, so that we can assign each person in the system a unique identifier. This will be used, in part, to allow access to the ‘home’, represented by out motor shield and continuous rotation servo. We imagine that the servo will be used to allow a door to unlock, giving the user access to the home. Upon entry, the home must conform to the needs of the resident. This is done through sensors found on the Sparkfun DangerShield, and allows for interaction with several LEDs that serve as an analog for a lighting system. All of the learning and processing occurs on the Arduino Uno. This is all mounted on a breadboard with the aid of leads.

**Code Overview**

1) PN532 reads RFID tag and either does or does not authenticate

2) Motor shield grants access to the ‘home’

3) Arduino Uno pulls and modifies the users home preferences and food become autonomously customizable through sensors and LEDs

**Reading the RFID**

The PN532 is an antenna that operates on magnetic frequencies. Using the magnetic frequencies emitted, power is transferred into the Mifare Classic RFID card. Upon transmitting the appropriate authentication key, the RFID chip responds with unique identifying information. Unfortunately, the libraries used to read this information and place them in a string has since been deprecated, and replaced by a library that does not suitably replace it. The best function we could find for this purpose was called ‘nfc.readPassiveTargetID();’, but this is a void function, preventing us from putting its output into any form of string. Its sole function is to read the contents of one of the sectors and then immediately dump it to the serial monitor, preventing this output from being interpreted or interacted with.

As a result, we used the built-in UID to help create a unique identifier for each card by running the UID (which can be found) and modifying it to become an ‘int’ in byte format, generating a unique string that can be compared to other strings. With this, we were able to identify individual cards, and as a result, individual users.

In order to do all of this, however, we needed to make sure that the communication protocol is clear both in the code and in the hardware. The PN532 supports both I2C and SPI, leaving the choice up to us. While SPI will be faster, we are dealing with such small amounts of data (the board would see at most 2 bytes) that this difference is negligable. The nuances of slave and master architectures that this would bring up becomes irrelevant because of a lack of other boards that data is being transferred between, so in the end, the deciding factor is that I2C is simply less complex. Unlike the 4 wires required by SPI, we would need only 2 by going with I2C. As a result of this advantage of simplicity and a negation of all the apparent downsides, I2C is clearly the better option.

**Automating Food**

OurAbode remembers and learns what you like about your food, and helps you in making better food in the future. Specifically, this prototype helps the user make toast, an easy to make proof on concept. As a result of time and resource constraints, we do not have the ability to fully automate a toaster, but our program will help the user in making the ultimate in toast. Upon learning what type of bread you have used, it times how long the user leaves it in the toaster and tells them when to take it out by randomly selecting a time within a range. The user then has their meal and rates the toast as either too much, too little, or good amounts of toastyness. If the rating is too much, the range moves downwards. If the rating is too little, the range moves upwards. Finally, if the range is good, the range tightens. In this way, the system will learn the preference of the user. However, this is also where the RFID system comes into play, as the ability to differentiate between users means that the system is able to keep track of individual preferences for individual users, furthering the ability of this system overall. While this is being used simply for toast, it could be expanded to apply to general cooking and kitchen tasks, or perhaps to broader tasks such as preferences in thermostat settings.

**Automating Lighting**

The OurAbode system also remembers and learns about your lighting preferences. The system allows for the user to manipulate 3 sliders to change the brightness of 3 different LEDs, acting as our stand-in lighting system. When the system is turned off the first time, it will remember the lighting settings that it held at that time, and upon the next startup, will resume to those settings. However, if the user decides to change the settings, the system will allow for this to occur, and will keep a record of what is being adjusted. The most common state will be stored, and will become the default of the next run.

**Storing Profiles**

In order for the entire system to operate, it needs to have profiles that it can pull for each user. Storing this information in the card itself would be difficult and would make the user take longer than desired simply to read the total contents of the card. As a result, it is much more efficient to have this information stored in a more readily accessible location, and rely on the RFID simply for identification.

The issue with this, however, is that the arduino does not, by default, have any way to store information after it has been turned off and turned back on again. The system is useful only if it works in these sorts of scenarios, so it is important to be able to save information through a power cycle. As we do not have access to the SD card Arduino shield, we needed to find another solution.

This solution was to use the serial data port. The arduino is already attached to a computer via the serial data port so that the programmer can upload new code, or the user can see messages from the arduino. Using the rxtx libraries, we are able to establish a connection between the Arduino and some Java code. Using custom function that we created for this purpose, the Arduino requests specific data. The java code on the host computer interprets this request and sends the appropriate data over. The Arduino receives that data and uses it in the program. This allows us to have complex profiles on our users that are both easy modifiable yet functional.

**Keeping your Home Secure**

It is an inherent weakness in typical smart homes that they are insecure. Any wireless form of communication, such as the ever-popular WiFi or Bluetooth that is so popular in these systems, is vulnerable to a wireless attack that could lead to a compromise of sensitive and personal data. Even if an attacker is unable to gain access to compromising data, there is still a potential for a threat. Automated lightbulbs, for example, can give an attacker knowledge about where an individual likely is in their home, or if they are home at all.

So while smart home systems have these inherent weaknesses, they don’t have to be so vulnerable. The OurAbode system keeps all data locally, and it will never see any network, preventing it from ever getting out. In addition, the cards themselves do not have any information save a string of hex characters used for identification. As a result, they do not have any information that can be manipulated to the detriment of the owner.

Even the simple manipulation of the cards has roadblocks for a potential attacker. The cards require an authentication key in order to read the data of the card, meaning that duplication is not possible unless the attacker either knows the hidden key or has some way around it. However, the authentication key is possible to change, so if the owner ever feels concerned about the safety of the card, its values can be easily changed as needed.

Finally, OurAbode keeps track of who is and isn’t inside the home as another safety feature, but this is not something that is ever shared to anyone who does not have a direct connection to the terminal of the program, by which point the attacker would already be inside.

Attackers will find this technology far more difficult than a standard lock, which can be easily bypassed mechanically or forced open. Given that the only way in is through the radio waves, OurAbode is a safe and secure system for the home.

**Obstacles we Overcame to Complete this Project**

As much as we hate to admit it, making this was not a process that went as smoothly as we would have liked. The first issue we encountered was the deprecated libraries of the PN532. While we were eventually able to find a solution, it took a fair amount of time to get to that stage, which cost valuable time. Figuring out how to store data was also an issue. We played with the idea of using a Raspberry Pi as a peripheral, but decided that that would be too complex and that although the solution of using the serial port for two way communication was also complex, it would be significantly more straightforward and usable for the time being.

Another, broader problem was our lack of overall experience. While we had all had experience with C or C++, we found that the Arduino derivative of C was different, and we were out of practice to the point that we found this to be a significant issue. Overall, it led to complications as we struggled to find more elegant methods of completing our tasks, and frequently had to spend time looking at documentation to attempt to discern why things were not working the way we had anticipated.

**Areas Where this Project can be Improved**

The OurAbode is a small scale prototype, so the most obvious manner in which it can be improved is to move to a full scale. The analogous door is the servo, so this would need to be replaced with a stepper motor or a motor with some form of potentiometer or motor controller so that the actual angle traveled can be measured. This would allow it to be mounted to a door and reliably act as a deadbolt. Of course, it would need to have some sort of failsafe in the event of a power failure. Electronic locks typically have some form of power storage for such an event, and we would need to do this as well. For our purposes, a set of large capacitors would work best, as they would be able to hold a charge for a large amount of time, would not expire, and would hold enough power for the low-power operation of reading an RFID system. Doing it in this way would mean no need for a mechanical lock, which would mean that there would be potential attack vector in the form of a mechanical weakness introduced by handles or lock bible systems.

Following this, the LEDs that are analogous for lights would need to be equated to an actual lighting system. This would be relatively easy to do by integrating into the pre-existing electrical system, by replacing the electrical switches with switches that are interconnected. As stated prevously, the ideal for security would be to stay wired, so it would be best to send signals by intentionally creating minor yet recognizable fluctuations in the power system. The signal integrity would be very poor, leading to a steep drop in communication speed, but it would still be viable as the speeds to not need to be particularly high to communicate information about which light is on and when.

Furthermore, better hardware structure is needed. The current design is a breadboard with various arduino shields mounted in various places, and if a chip was designed with the goal being specifically used for this purpose, it would have a greater efficiency

At the moment, the system requires a host computer for interaction with the main board. Ideally, this would not be needed, and a user who had a need to configure things could do so by interacting with a port, rather than having a computer held hostage to see that it will work.

This also connects to the need for higher memory capacity. If an SD card shield were to be mounted, then it would be possible to read and write to memory in a much more straightforward manner, as it would no longer be necessary to interact with the rxtx libraries to establish a connection over the serial port to Java code. This would also aid in removing the host computer, as it would not longer be needed for memory storage.

Another potential area for improvement would be to integrate NFC technology. NFC is being found on more and more phones and smart devices, and uses the same protocols and frequencies as RFID technologies do. As a result, it would be viable to have a phone used as the RFID tag with the aid of an app.