ABT Smart Home Report

CS122A: Fall 2019

Angel Renteria

Brandon Tran

Table of Contents

Introduction

We decided on creating a smart home system as a way of being able to implement a variety of wireless communication methods within our project. The idea of creating a smart

home system was interesting in the sense that there are a variety of different components that make up a smart home. In this sense, we can really implement the techniques and methods of communication that we have been taught and put them into practice. Along with being able to work outside of our comfort zone by introducing new technologies that we have not worked with previously.

As explained above the smart home system for our project will compose of a few different components. Firstly, the smart home system should allow for motion detection sensors to trigger lights within an area of the home. The motion detectors will communicate using Zigbee to create an area network controlled by a home station. Secondly, the home station should notify you when motion is detected while away from home. This will be done via wireless communication between the home station to an application on the web. Thirdly, we will have a TFT LCD display connected to the home base (a microcontroller) that will allow the homeowner to control the system. This is done by displaying a menu with a variety of options such as light adjustment, connected devices, and a “find devices” option to add additional lights. Lastly, we want the lights connected to the system to be dimmable. The home station will control the level of dimness that the lights should be set to.

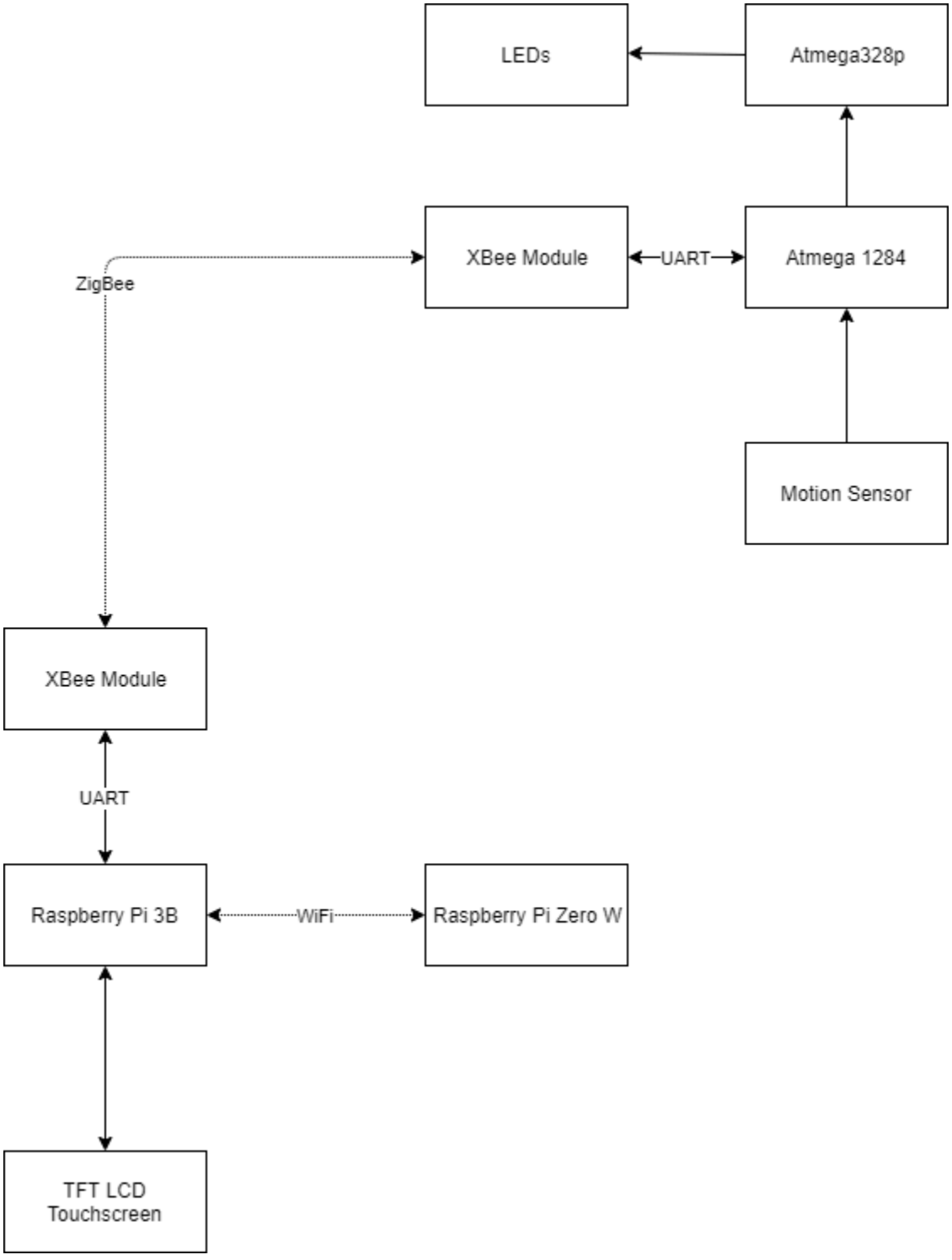
Hardware

Parts List

The hardware that was used in this project is listed below.

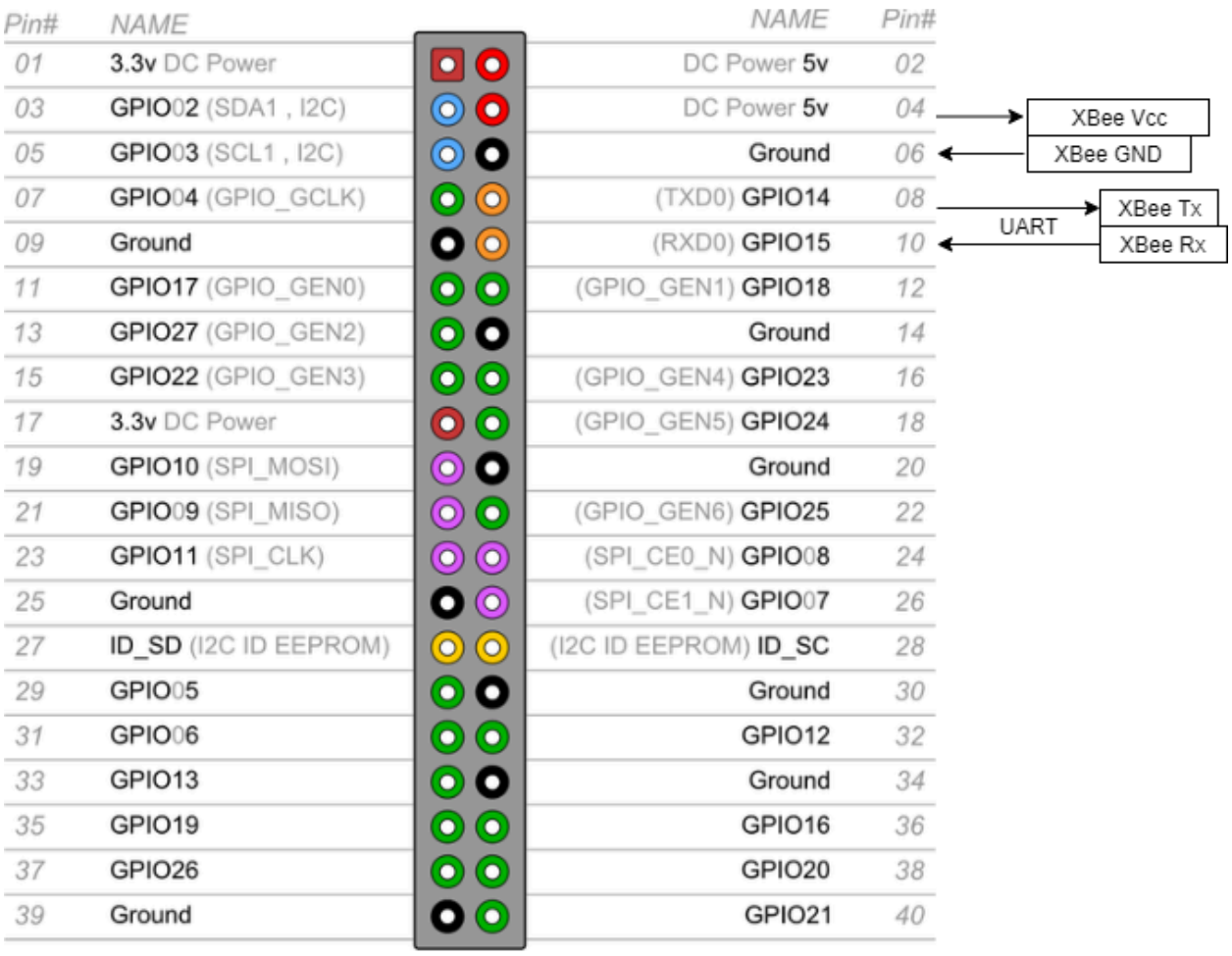
Part	Part #	Quantity	Price (optional)
ATMega1284	ATMega1284	2	$5.00
ATMega328p	ATMega328p	2	$2.50
Raspberry Pi	Zero W	1	$26.99
Raspberry Pi	Model 3b	1	$35.00
Xbee Module	XBee S2C 802.15.4 RF	3	$26.95
Pir IR Sensor	HC-SR501	2	$1.89
7in TFT LCD Screen	MPI7002	1	$49.99
USB Serial Port to Xbee Serial Adaptor	FT232rl	3	$8.99
		Total	238.58

Block Diagram

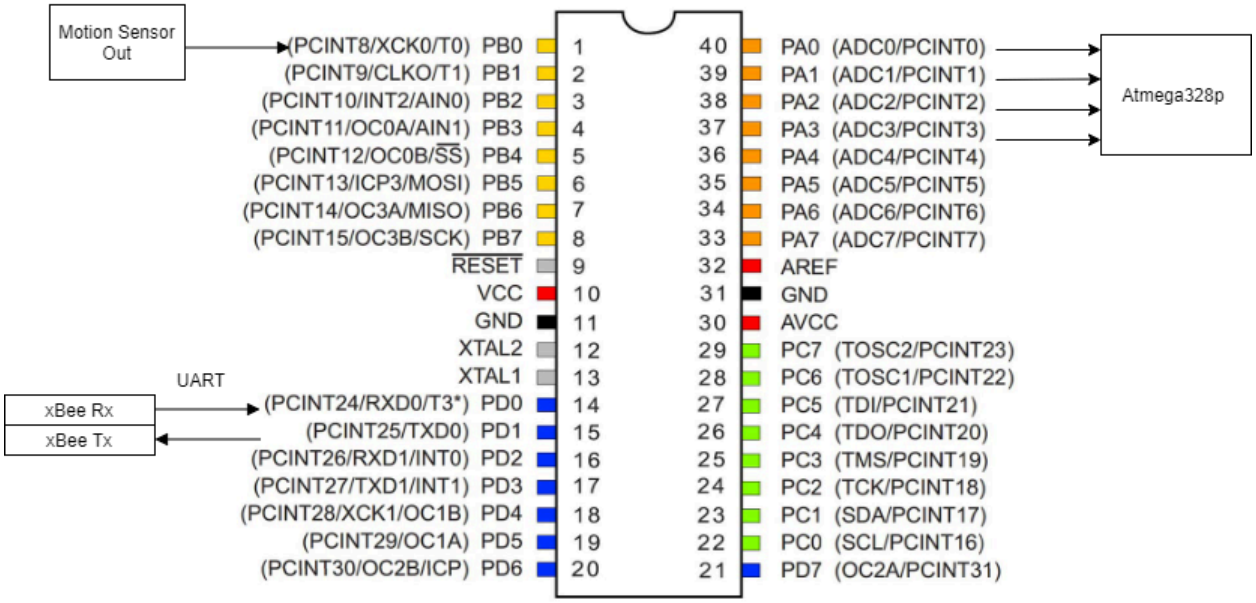


Pinout

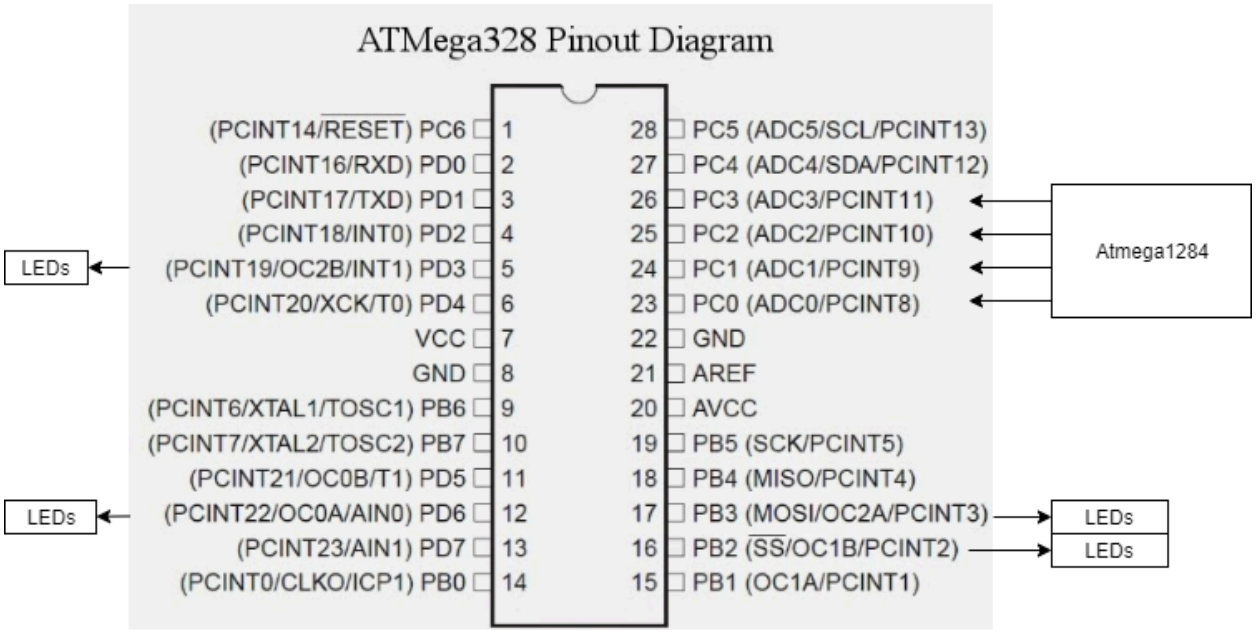
[Raspberry Pi 3B Pinout]



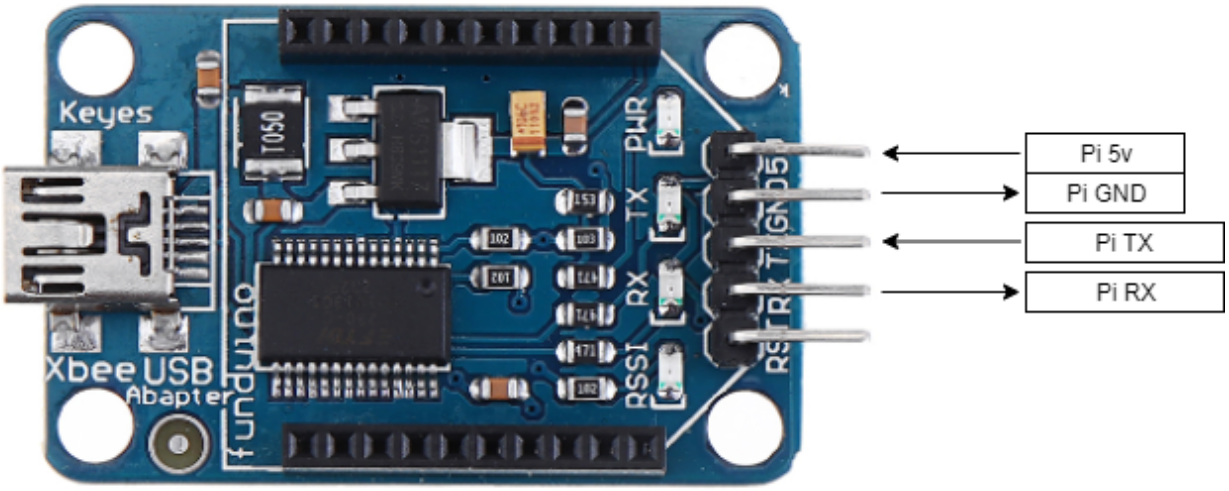
[Atmega1284 Pinout]



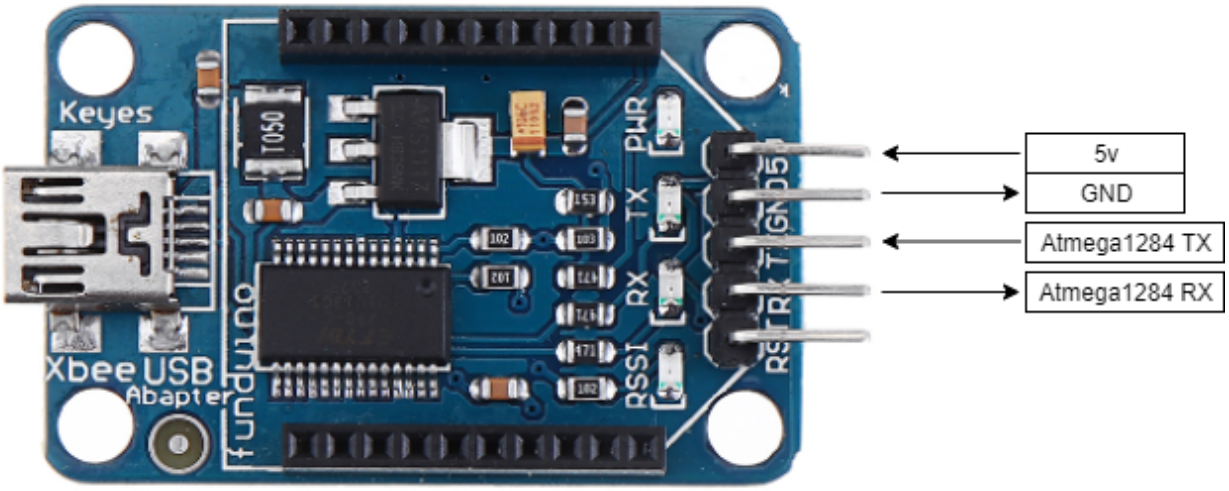
[Atmega328 Pinout]



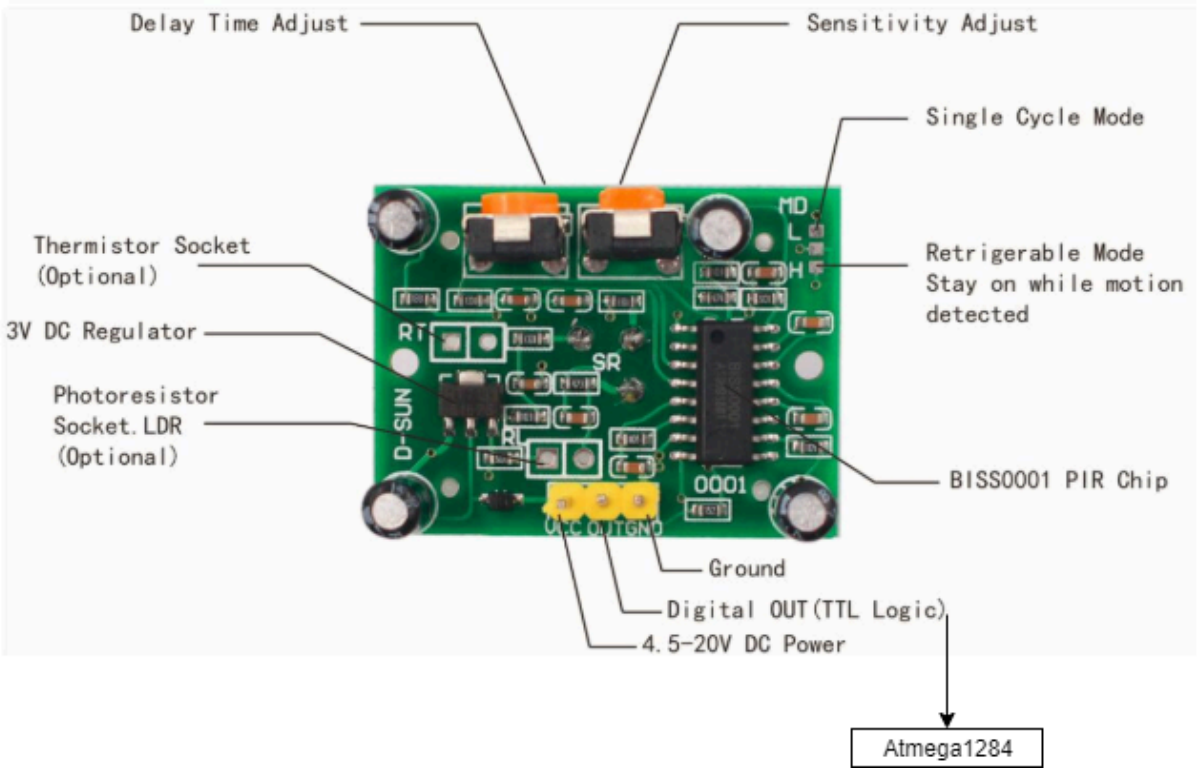
xBee Breakout Board to Pi



xBee Breakout Board to Atmega1284

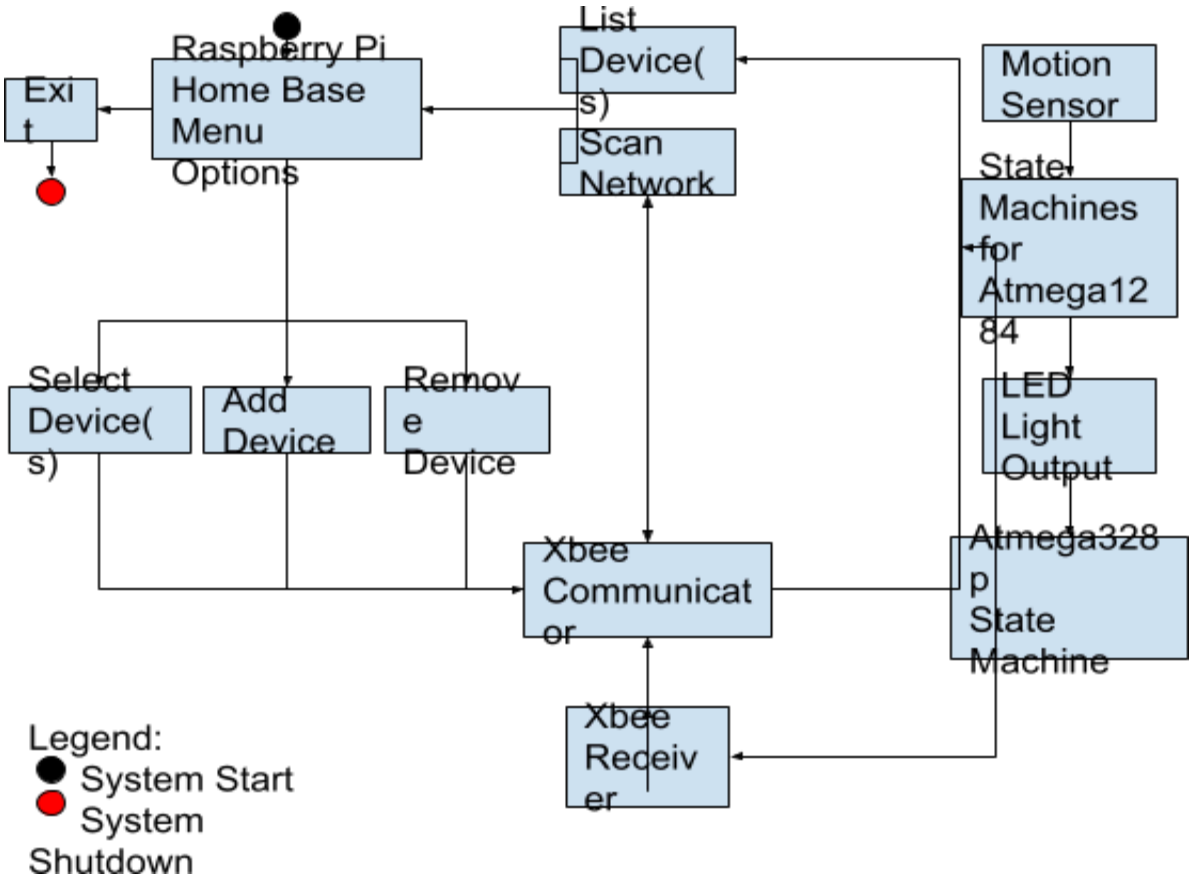


IR Motion Sensor Pinout



Software

The software designed for this project was implemented using the PES standard. The overall design as a task diagram is included below.



Raspberry Pi Home Base Menu Options - The home base system displays the menu options through an LCD display and is ready to send information to a server once external nodes are connected to the system.

List Device(s) - The list device task gets information about the current network from the Xbee communicator. It displays the current network using a 64bit address for a node followed by an identifier that was given to a node.

Scan Network - The scan network task can begin the search for nodes present in the surrounding area. It does this by broadcasting to those devices on the same local network and keeping a list of those devices found during the scan.

Select Device(s) - The select device(s) allows for the selection of specific nodes in the network to interact with. Allowing for mode selection of lights on, lights off, and lights that trigger using the motion sensors. In both the lights on and motion sensor modes, the lights can be altered using a brightness setting.

Add Device - The add device task can initiate a search for a node given an identifier to search for. The node must have been configured to connect to the same network as the communicator to be picked up.

Remove Device - The remove device task can begin the removal process of a node from the network. We remove the node from the network list by sending it data through the communicator and ensure that data won't be received from the specified node through the ATMega1284 state machines.

Exit - The exit task beings a shutdown of the system. It clears the network list, as well as disabling information received from the nodes. Finally, the interface is shut down once all the clean up has taken place.

Xbee Communicator - The communicator manages the local network that is communicated to the home base. It also allows for wireless communication to the home base and external nodes in the network.

Xbee Receiver - The receiver plays a smaller role in that it manages communication from the home base to the current node it is apart of. Allowing for information to be sent and received network.

Motion Sensor - The motion sensor task triggers upon detecting motion. At which point an interrupt on the ATMega1284 triggers and begins to emit lights from LEDs connected to an atmega328p.

State Machines for ATMega1284 - The state machines in the ATMega1284 focus on observing the room they are sent to monitor. This includes triggering lights based on a pin interrupt, receiving instructions from the home base, and sending data about motion detected to the home base.

LED Light Output - The output task on the ATMega1284 is what allows for lights to display on the ATMega328p.

State Machine ATMega328p - The state machine for the ATMega328p interprets the output from the ATMega1284, and sets the value of brightness on the LEDs based on that value.

Implementation Reflection

We are glad to have been successfully completed our proposed project. It was a great experience to be able to tackle new hardware, languages such as Python, HTML, Javascript, and lastly to have been able to continue to build experience working on a team. The process as a whole really allowed us to come together and implement a smart home system that from this point can only be refined, or continually built upon. That is not to say that there were some problems along the way.

One major thing we would change if we had to this again would be the initial planning for the project. While we had an idea of what we envisioned for the project, the problem was that we never really took the time to verify and research the parts themselves. This lead to a lot of headaches and confusion about what certain parts would do, or have to change how certain aspects of the project were going to be implemented. Having a better idea of what parts could have been used, or if they would actually allow us to implement our design would have saved time allowing us to work on other aspects of the project.

Looking back on the process as a whole, it was nice to see that we were able to communicate to resolve issues in the design process. Whenever we ran into a problem, we would ensure that we could resolve the issue before moving forward. That along with creating smaller test programs for hardware gave us the ability to have a solid foundation that would make creating aspects of the project come together nicely. It was really nice to see everything piece together near the end because we really understood how everything interacted with one another successfully

Milestone

Milestone I

This is what we had planned to do for our system:

- A simple base station that can connect to a single microcontroller/motion detector combo.
- The base station should acknowledge that this device is connected, and receive information about motion detected in an area monitored by the sensor
- Zigbee communication should be implemented to allow intercommunication between the base station and external microcontrollers.
- The motion sensors should be able to detect movement and interrupt the atmega1284 to notify movement via a pin change interrupt.

‘

We were able to create a simple base station that could connect to the sub-controller(atmega1284), however, it was not connecting as we were expecting. We were unable to make the uart on the Raspberry Pi communicate directly to the XBee module to get the correct values. As a result, we managed to demo this by using an intermediary Atmega1284 to communicate between the xBee module and the Raspberry Pi. This allowed us to complete the other checkpoints: that the base station acknowledged the sub-controller and receive information from the sub-controller; Getting Zigbee communication between the base station and the sub-controller. We had difficulty demoing the motion sensor because the sensor that we had gotten initially was too sensitive to the motion in the lab rooms, but we were eventually able to demo this checkpoint towards the end of the lab. We felt like more could have been done if we were able to fix the problem of our understanding of UART for Raspberry Pi.

Milestone II

During our second milestone, the project should implement these features

- The home base should be able to detect multiple external devices and connect with them.
- The home base can interact with a TFT LCD screen. The LCD screen will have a menu that can control aspects of the system such as connecting and removing devices and displaying active devices.
- A simple android application should be implemented that can connect to the base station for future communication.
- The WiFi module should be implemented to allow for wireless communication between the base station and android phone.

We were able to detect and connect the base station to multiple devices and communicate with them. During this time, we fixed the problem that Raspberry Pi was able to directly communicate with the XBee module. We were not able to create a simple android application to communicate to the Raspberry Pi due to time constraints, so we made an alternative web application to demo for the milestone. We were able to demo the interaction of the home base on the LCD screen, but not what we were expecting due to some hardware problems. We didn't have a WiFi module for allowing the wireless communication between the base station and android phone because the Raspberry Pi's internal wireless capabilities were sufficient enough to demo. As a result, we were able to demo the milestone but nearly fell short due to roadblocks from hardware failure and time restraints.

Completed components

We have completed the proposed project. There were changes to some of the components in order for us to complete it on time. We had changed from an android application to a web app. The core functionality of the project works well as the user is able to interact with the smart home system via getting notifications and changing the setting of the motion-sensing smart lights.

Incomplete components

One aspect of the project we didn't complete was the creation of an application for a mobile device. We had initially proposed that we would have an application that would have received notifications about motion being detected. Due to time constraints, we never actually made the application. At the time, we were working in our development phase trying to apply the XBee module to the base system, along with creating an interface for the home base. The role of a phone application in this project really didn't really make sense if we were going to solely implement notifications. In the end we decided to implement a web app that would handle the notifications from a server that we created. This could have been avoided if we spent more time with the initial proposal.

Youtube Links

https://youtu.be/KwTkpdIE-WU - Short Overview of the project

https://youtu.be/y6XVFFTZl9k - In Depth Review

Testing

Motion Sensor:
We created a small program to test the simple flashing of an LED using a pin interrupt. This allowed us to verify that this aspect of the system was working as intended before moving on to the nodes. We did initial testing at home, with other roommates walking around the apartment. However, what we should have done for further testing was bring this into an area with a lot of activities. This would have let us know ahead of time of the issues that the initial sensor we used was too sensitive for the project.

Xbee Modules:
We made simple programs on both the ATMega1284 and Raspberry Pi 3b that ensured that we could have wireless communication between the two devices. We initially ran these on the ATMega1284, but it took us time to figure out the UART problem before we could implement the Xbee on the Raspberry Pi Zero W.

ATMega328p:
We wanted to ensure that we could use the various PWN channels on the 328. We did this by implementing a simple program that tested the PWM dimming functionality. Doing so allowed us to understand how the PWN functionality could be implemented and integrated in the project as a whole.

Flask Server:
To ensure that the server was receiving the data from the base station, a mock base station was created to generate random requests to the server. The test would then print out the values on both the base station and server.

GUI:
When it came to testing the interface we had for the system, we built each piece of the menu up piece by piece. We had our roommates interact with it the various menu options. Eventually, we once we could ensure that the menu was functional, we integrated the base functions to the external nodes they were connected to.

Nodes:
When it came to testing the nodes we already had smaller aspects of the system already tested. All we had to do was create the state machines that would integrate those systems. However,

we didn't test this part of the system as extensively as we wanted to. Ideally, we wanted to implement simulations of the system. Though this meant that we could pinpoint issues in the system faster since we knew certain aspects of it were already working as intended.

Web App:
The web app will display the latest 4 updates from the server. In order to test this, the server would run a loop in the background that will update a value every second. The web app should display the values that the loop will print in the terminal.

Known Bugs

- Bug: When the motion is detected over a long period of time the lights will blink off for a small period of time
  Debugging: Check if the state machine for the motion detected keeps the light on as long as motion is detected.
  Possible Fix: make the state machine that deals with the motion detected reset the counter for the light every time motion is detected.
- Bug: The menu for setting the smart light does not change the value on the gui to 15 for on and 0 for off
  Debugging: print out to the terminal to show the value as the option is selected in the menu
  Possible Fix: See if selecting the options on and off can update the value of the dimness setting in the text box.
- Bug: When scanning the system again for nodes, sometimes a node that was already on the list is duplicated
  Debugging: print out the values that is in the list
  Possible Fix:: Compare the nodes that are already in the list to the nodes that are being added and only add the nodes if they do not match to any currently in the list
- Bug: The pwm on PB2 will always be on even when set to a value of off
  Debugging: Turn off the pwm on the pin and see if the pin itself is working correctly by blinking a led.
  Possible Fix: change the pwm pin to a different pin.

Resume/Curriculum Vitae (CV) Blurb

ABT Home System
Collaborated with another individual to implement a smart home system. Learning new languages such as Python, Javascript, and HTML to create a web application that receives information from a server. Incorporating UART functionality from the ATMega1284 and Raspberry Pi 3B to use wireless

communication via XBee modules using the ZigBee IEEE 802.15.4 protocol. Using Python to create a user interface to interact with the smart home system.

Future work

If we were to continue the project, another feature that we would add is to have the web application be capable of modifying the smart light settings over wifi. We will also build a case for the base station and the base station to make a virtual keyboard to allow for text input on the menu. We would probably have a custom PCB made for the smart lights/sub-controllers to reduce the size of the system. The PCB will then fit into a casing that will allow for power to be taken from a standard light fixture, to allow for compatibility in homes. We could also make the base station to group smart lights so that the user can perform the same settings to multiple lights.

References

If you used code or ideas from another source (including inspiration from a video or another product) please list those sources here. Include the license for the code that you used or referenced.

Link to all the resources used: Project Resources

Appendix

Include images of all of the SM's that you have designed and any other work that you think is relevant to this project.
Link to the github: https://github.com/B-Tran/CS122A-FinalProject
Link to state machines: State Machines