A

Project Name: Smart Garage
Course: EECS 129B
Professor: Klefstad, R.

Done By: Nomer Reyes
Andy Cheng
Angeline Halim

Abstract

The project involves the design of a smart garage door opener. The garage door will open when all requirements are met. A microcontroller processes the information received from the different sensors mounted near the garage before activating the motor that controls the garage door. We designed, built and tested this system to its successful operation. This report explains in detail our design and test results.

Project Description:

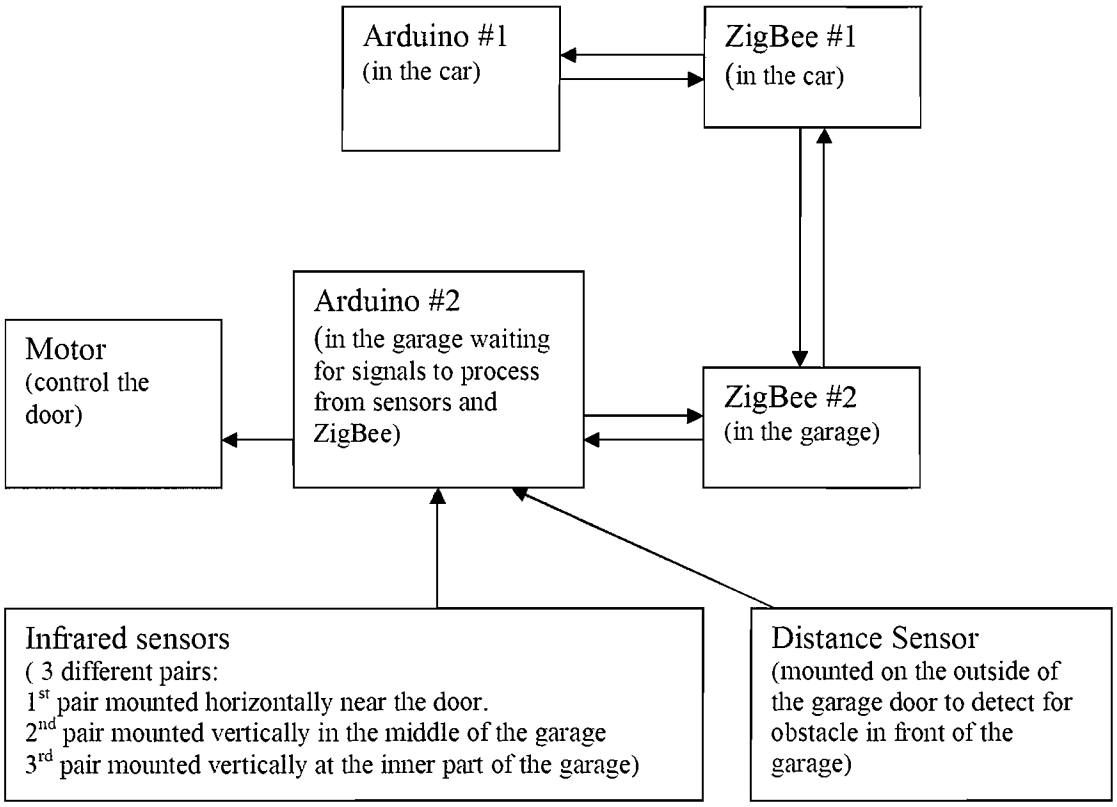
Purpose – The objectivity of this project is to improve the existing garage opener design as well as to apply the wireless communication techniques that we learned last quarter.

Background – We are bothered by the fact that every time when we want to open the garage door, the garage opener is either borrowed by other family members or is some other cars. So to change this, we came up with the idea that to have the car communicate with the garage system itself without having to go through another device (the garage remote controller). Even though this project might not be the leading technology right now, it is very useful and practical for everyone.

Features – The main part of this project is the sensors that we use. We have adopted a few sensors and they are Inferad proximity sensors and Inferad Octo-couplers. The proximity sensor, which is placed on the garage door, will detect and alert the garage system once an object is in the effective contact range with the door. The garage system then wirelessly prompts the approaching object for password to gain entry access. The Octo-couplers, which are placed inside the garage with two pairs, will detect if anything is blocking the garage door or if the car is parked in spot. If there is something that is blocking the garage door, the system will halt closing the door until the object is removed and the door is safe to close. And if the car is not parked in spot, then the system will also halt closing the garage door until the car is parked in on the spot as where it is supposed to be. One more interesting thing that we will use for controlling the door is a stepper motor. Depends the code, the door can rotate either counter clock wise or clock wise.

System Configuration – The system will require a stepper motor, two pairs of octo-couplers, one proximity sensor, two arduino board, two Xbee chips, two major breadboard, a mock garage door with walls

System Level Block Diagram



Arduino #1 (on the RC car)

When it detects the other arduino, it will wait for a signal so that the ZigBee #1 will be able to send data to ZigBee #2.

ZigBee #1 (on the RC car)

When signals are received, it will send information to the microcontroller (arduino #1). After verification, arduino #1 will send data to the ZigBee #1 so that it will be transmitted wirelessly to ZigBee #2.

Arduino #2 (in the garage)

At first, it will be placed on a standby mode waiting for data to be received from the other arduino. When it is activated, it will process data received from the sensors and zigbee.

ZigBee #2 (in the garage)

It will first send a series of code to the ZigBee #1 for verification. Next, it will be in receiving mode waiting for signal from ZigBee #1. When signal is received, it will notify the microcontroller (arduino #2).

Infrared sensors

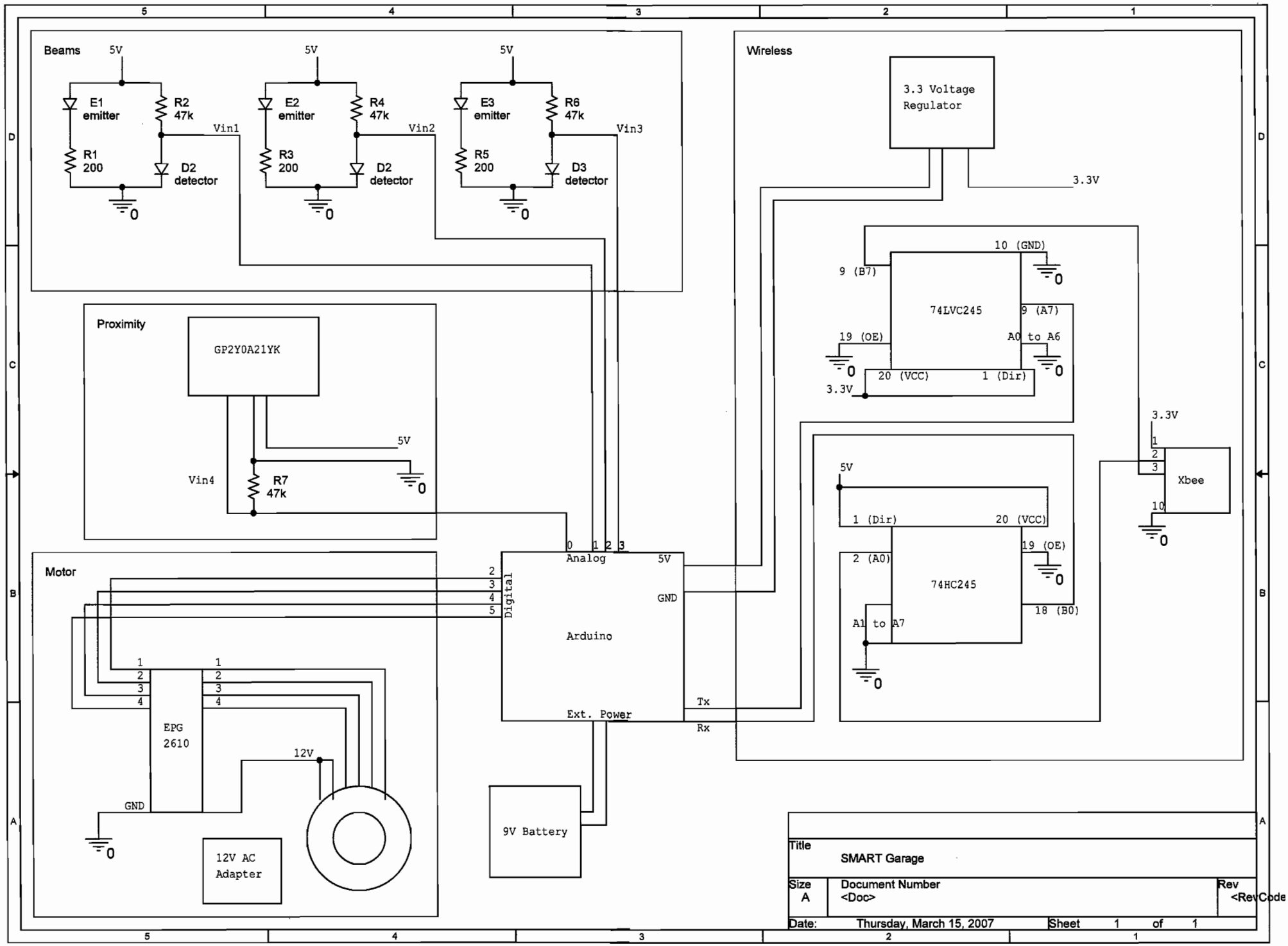
Several pairs of light emitters and detectors are attached to the inside of the garage for obstacle detection. There will be one pair that will be mounted parallel(horizontally) to the door to make sure that the garage door will not close on any obstacle in the way of the sensor beam. Two other pairs are mounted in the inner part of the garage. One will be mounted vertically in the middle of the garage and the other vertically at the inside most of the garage to determined if the car is already inside the garage. When all conditions are met, the microcontroller (arduino #2) will send signal to the motor for activation.

Distance Sensor

An infrared device used to detect obstacle reflectivity and ambient light. This device will be attached to the outside door of the garage. When it detects an obstacle within the specified distance, it will send information to the microcontroller (arduino #2).

Motor

Motor will be activated when a signal is given from the microcontroller (arduino #2). Arduino #2 will send a signal to the motor when all conditions are met depending on the position of the RC car.


Beams
5V
E1 emitter
R2 47k
R1 200
D2 detector
Vin1
E2 emitter
R4 47k
R3 200
D2 detector
Vin2
E3 emitter
R6 47k
R5 200
D3 detector
Vin3
0
Proximity
GP2Y0A21YK
5V
Vin4
R7 47k
Motor
EPG 2610
12V
GND
12V AC Adapter
9V Battery
Analog
Digital
Arduino
Ext. Power
Tx
Rx
Wireless
3.3 Voltage Regulator
3.3V
74LVC245
10 (GND)
9 (B7)
9 (A7)
19 (OE)
A0 to A6
20 (VCC)
1 (Dir)
74HC245
2 (A0)
18 (B0)
A1 to A7
Xbee
Title
SMART Garage
Size
A
Document Number
<Doc>
Rev
<RevCode
Date:
Thursday, March 15, 2007
Sheet 1 of 1

Software Description:

Sensors:

The software will be is quite straight forward. And we will start the description with the proximity sensor. When the proximity sensor detects an object within 80 cm, its internal resistance will go up as the object gets closer. When the resistance (read value) is over 230, it will first prompt the activate the arduino board in the garage system and it will prompt for a password. Another sensor is the octo-couplers and it used in the similar way. When the read value is over 900, the octo-coupler will alert the garage system arduino board to halt every action.

Proximity Sensor:

```
if(value > 230)
  promptPassword(); //this will start prompting for password and activate the Xbee
  digitalWrite(GARAGEdoor,HIGH); //this will open the garage
else if(value < 220)
  digitalWrite(GARAGEsys, LOW); //this will keep the garage closed and Xbee down
```

Octo-Coupler Sensor:

```
  if(value > 900) //beam
  {
    haltAction(); //stop any executing action,
  }
  else
  {
    resumeAction();//continue any actions
  }
```

Stepper Motor:

The stepper motor is controlled by allowing current going through the desired transistors and creates a magnetic field that controls the motor movement. We use a code like the following:

```
digitalWrite(outputPin1, HIGH);
digitalWrite(outputPin2, LOW);
```

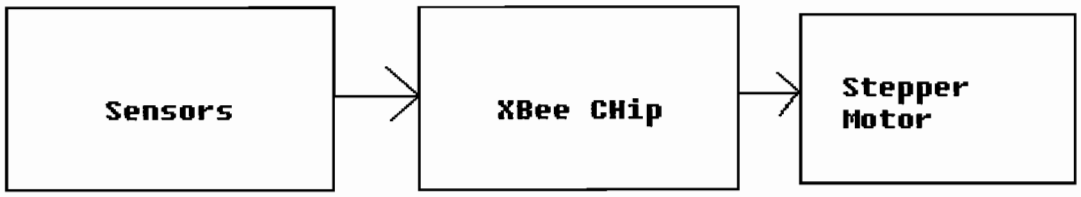
```
digitalWrite(outputPin3, HIGH);
digitalWrite(outputPin4, LOW);
```

This is an example of how to control a stepper motor. When we allow the outputPin1 and 3 to be high, it will make the pointer in the motor to point at 12 o'clock direction. When we switch the outputPin 2 and 3 to high, it then move the pointer to about 3 o'clock and the motor is rotate clockwise for about 90 degrees. We will have this code that goes both clock-wise and counter clock-wise.

Xbee Wireless:

In this section, the main part is just to set up the password and activating the Xbee. To save power, the Xbee in the garage system will not be activated until the proximity has detected something. And the Xbee in the car system will not be constantly sending password either because that will drain out of the battery. We will have the Xbee's communicate with each other only when the sensors are sending signals to activate.

Flow Chart:



The sensors activate the Xbee chip which activates the motor

System Test Plan

Conditions	Results
1) Car is on the driveway	The garage will open. The proximity sensor detects the car on the driveway. The system will request the password from the car module. This module then sends the password. If the password is successfully verified by the system, the garage motor turns on.
2) Car is out of range	The garage will not open. The proximity sensor does not detect the car and therefore the system will not ask for the password. The module on the car will not be prompted for the password.
3) Car is out of range, but some random object is detected by the proximity sensor	The garage will not open. The proximity sensor will detect an object and the system will send a request for the password. However, the car is out of range and will not receive this prompt.
4) Car is in range, but not on the driveway	The garage will not open. The system will not request the password because the proximity sensor does not detect an object in the driveway.
5) Car is on the driveway, but the password is wrong.	The garage will not open. The system will detect the car in the driveway and request the password from the car. The car will send the password, but the system will not open the garage with an incorrect password.
6) The garage is open; the car enters the garage and parks.	The garage door will close. The system will detect the car pass into the garage with an optical sensor. Then the other two optical sensors will determine that the car is parked properly and the garage will close.
7) The car enters the garage and parks, but there is something blocking the garage door.	The garage door will not close. The system will determine the car is parked properly, but the first optical sensor will be blocked, telling the system there is something in the way.
8) The car enters the garage and parks, but it is not parked correctly.	The garage door will not close. The system will not be able to correctly determine if the car is in the garage.
9) The car is in the driveway and it is night.	The garage will open and the lights inside and outside of the garage will turn on. The photoresistor will determine that it is night and the system will turn on the lights.
10) It is night but the car is out of range or has the wrong password.	The garage will not open, but the photoresistor will determine it is night and turn on only the outside lights.

Part list

PART NAME	PART NUMBER	QUANTITY	MANUFACTURE	COST/QUATITY
Arduino	Arduino-USB	2	Sparkfun	$31.95
ZigBee	XB24-AWI-001-ND	2	Digikey	$23.20
Breadboard	194271	2	Jameco	$15.95
Infrared proximity sensor	GP2Y0A21YK	1	Sparkfun	$11.95
Octal bus tranceiver	74LVC245	2	Digikey	$0.60
Octal bus tranceiver	74HC245	2	Digikey	$0.58
Voltage regulator	497-1491-5-ND	2	Digikey	$0.77
9V Battery	N/A	2	Jameco	$0.34
Switch	315580CM	2	Jameco	$1.35
Wire	100' reel 22awg	1	Jameco	$5.49
RC car	N/A	1	Target	$9.99
Stepper motor 12VDC	42M100B2U	1	Digikey	$18.70
12V power adapter	2731776	1	Radioshack	$17.99
Infrared diodes	TLN110JP	6	Jim-pak	$0.65
Resistors	N/A	10	jameco	$0.01
TOTAL				$217.62

Summary

In our project, we faced many problems due to the multiple components we had integrated together. The first problem we encountered was the motor. We were using a regular motor at first but found that we were unable to precisely control the starting and the stopping of it when pulling on the garage door. We obtained a stepper motor that was more powerful and more precise. A problem we found with the stepper motor was that it required more power than the Arduino could provide. The Arduino did not supply enough current to power the motor, so we purchased a 12V DC adapter to use solely for powering our motor.

The next problem we faced was with our wireless system. Using a design that was provided for us from EECS 129A, we created two identical circuits that consisted of an Arduino and a Zigbee chip. However, it seems that sending and receiving consumes a lot of power, as we found ourselves constantly having to replace the 9V batteries that were powering the system. Overall, we had gone through about 20 9V batteries in the process of building this project. Due to the nature of the wireless system, the sending and receiving would fail due to the low voltage on our batteries. One solution we tried was to use an external 12V – 600mA DC Adapter. However, after only about one minute the Arduino became very, very hot. As a result, we reverted back to using batteries.

Writing the program for our garage system also gave us problems. Since we had several components in our system, we wanted to write separate functions to allow each component perform its task. However, the sensors would not properly return their values when they were polled from a function. Also, since the Arduino essentially loops through a loop() function, so using loops within this loop() function was not a good idea.

These nested loops seem to cause timing problems because each component had to wait while another component was busy. Because we wanted each component to be functional at any given time, we had to refrain from the traditional sequential programming techniques and instead used states to keep track of the various components. We kept track of the state of each component using variables and performed actions based on what state each component was in. This way, the program could loop through each component and perform its task when necessary.

The rest of the components didn't really give a lot of problem. Our distance sensor performs accurately. Our tests proved that it performed within the specified distance every time. Extreme sensitivity of the infrared sensors emitters to its detector with respect to the surrounding lighting could result in significant errors. But these errors did not affect our overall result. We did accomplish our goals.