The Garage Backup Alert System

Philip D. Geramian
497307163
CSE-671
Syracuse University
Spring 2016
May 6th, 2016

Abstract:

In this paper I discuss the final project that I undertook for CSE-671 this semester at Syracuse University. For this project I made a garage backup sensor that would make it easier for the user to park their car with the front facing out, making it easier for them to leave with the car the next time they need to use it as they would just have to pull forward. The system is made up of a wall mounted base unit that contains a distance sensor, a BeagleBone Black, a 2-digit 7-segment display, and a wireless transmitter. The system also contains a wireless unit that sits in the car and provides both a visual representation of the distance the car is at (a red and green LED) as well as an alarm for when the car gets to the correct distance so the driver can know to stop backing up the vehicle. The project was a success in my opinion as all of the components of the system were able to work with each other, and the system functioned as desired.

The Garage Backup Alert System

Introduction

For those who drive a car everyday and park them in a garage they are faced with a daily decision every single time that they park, and most have no idea that there is even a choice. Park in, or Park out? What this means is does one pull their car straight in and have the front of their car facing in, or does one back up into the spot and have the front of their car facing out.

For most people the choice they take is to park in, because the thought in their head is that backing up is a pain, and on modern cars with their larger A-pillars than older cars this can be true, and is a reason for why backup cameras have become so popular, even on basic trim package, and come 2018 it will be federal law that all new cars sold must have this feature. But even with this feature, many people find backing their car in a hassle.

The disadvantage to parking in is that, all one is doing is putting off driving the car in reverse until the next time that you need it. Think about it this way, when is it usually more important that you are able to move quickly, when getting home, or when leaving home. For most, it would be when leaving home, as it is not unlikely that one is running late in the morning.

For this project I decided to make a device that would help people back their car into a garage, thus allowing the option of parking out to be easier for those who have trouble with driving in reverse. The most common issue with parking out is when do you know that you are the correct distance from the wall. Too close and you can not walk around the back of your car, you can't access the trunk of your vehicle, you bump into the rear wall and damage the wall, or even damage your car. Too far and you may not be able to walk around the front of your car, or your garage door will not close because you are blocking the safety sensor, thus keeping the garage door open.

What I decided to do was make two items that will work together to let the driver know they are the correct distance from the wall, a unit that gets mounted on the wall to measure the distance they are from it as well as print out the distance, and a small

remote like unit that would sit in the car and inform the driver by both a visual warning (a red LED) that they are too close or not (a green LED), as well as an audible alarm that will sound off when they get too close. As this unit is to sit in the car, it could not be connected via a hard wired connection and therefore had to be wireless. Throughout this project I was faced with multiple issues, but due to my skillset in debugging I was able to fix these issues as they came up.

Background

As stated before, new cars will be required to have a backup camera on them come 2018, but this is an option that can be seen already on a lot of cars, mostly as a point of convenience for the driver. But predating this car companies did at one point make detection systems that would alert the driver if they got within a certain range of an object. This system was largely done to help people be alert about cars that are around them, are they too close to a car in any direction was the key task. This system could also be used to assist in parking, but it would be most helpful when parallel parking as the distances the sensors were set to alarm at were fairly close.

A big issue with that system was that it had to be fitted at the time of manufacture, and that a lot of sensors had to be placed around the car so that the car could detect itself and where the other objects were. Also these systems were built very much into the cars, and every time you got a new car you would have to pay for the feature again, so the expense of it is only one time per car, not one time for life.

A feature of the system that I designed is that once it is installed into the garage that it is needed for, you can use it on any car that you want to. No special electronics have to get fitted; it does not matter if you purchased a new car tomorrow, just take the remote out of your old car and place it in the new one.

It can also be seen that with this system it could become a feature of the house for when the house goes on sale to the next owner, the listing could say “Has a garage with backup assistance installed inside.”

While I cannot find exact cost on how much it would be to install a sensor system to a car, I was able to find out how much a backup camera system cost, and the parts alone are on the order of $400. This not only makes installing this yourself using a factory kit hard to do, as the wiring may need to be run, but also fairly expensive too.

Methodology

For this project I followed the design method of build individual systems and test them, then once everything has been tested on its own, to then integrate them into one system, again doing testing along the way to make sure each part works in the system the way I had hoped for. To look at it a different way, my system can be divided into three subsystems that then get integrated into one large system to become the device I designed. The three subsystems are as follows: Display, Distance sensor, and Wireless system. To go a layer deeper the wireless system can be divided into two sub systems of its own: Transmitter, and Receiver.

This is how I went about in designing my hardware systems, but for my software systems I did things slightly differently. As I used the BeagleBone Black as the main part of my system I first took the time to determine what communication methods I would be using. As it turned out I would only use two communication methods, UART and SPI. Going from there I then went about writing libraries for these communication methods using C++.

To write these two libraries I had to spend some time looking at the way the Linux Kernel handles these, and as it turns out it is done via special registers that have definitions for them in special Linux only libraries. Once I understood how to use these features I was then able to write the libraries for both UART and SPI.

Once the libraries were built and tested I then moved on to writing the needed code to have the devices that make use of these communication methods being able to use them.

To help the process of testing each subsystem in a quick way I decided to do a lot of my testing of the subsystems on an Arduino as it is a platform that is very easy to write a

bunch of quick code for. A good example of this was the wireless system, as I knew that the remote I was going to build would run off an Arduino anyways I decided that for testing I would run the transmitter off of an Arduino too.

A subsystem that I did not use the Arduino for testing was the Display. The display that I used I had designed myself and had made it to run off of SPI. Since I had designed this display earlier this year for use on another project I knew that I could already communicate to it using the BeagleBone Black over SPI, and I also knew the methods that I needed to take in order to correctly send data to it such that the display would correctly output my desired values.

The tools I used in my design process were fairly minimal, an Apple MacBook Pro, a Fluke 17B+ multimeter, a FTDI USB to UART bridge, and a Digilent Analog Discovery. These tools make up my regular toolkit for when working on hardware systems and as this project involved making a hardware system, it was not a surprise to me that I used these tools and that I used them in the way that I did.

Each of these items plays a role in my design process and I will explain their purpose to me. The Apple MacBook Pro is my main computer, and has been for about 5 years now. This thing will not die even after all the wear and tear I have put it through, (4 years of undergrad as an engineering student, 1 year as a masters student, 5 years as a live production technician for both lighting and sound) and I am ever thankful to it for all it has done. The Fluke 17B+ multimeter is actually a relatively new addition to my toolkit, but the spot of a multimeter isn't. Previous to owning this fantastic tool I was used to using cheap, no-name multimeters that I was never quite sure how accurate they were, and how long they would last before they failed me. Also they ate batteries (9v cells at that!) at an impressive rate, and due to the cheapness of the meters, and the expense of the batteries I would sometimes just replace the meter as opposed to replacing the battery. Multimeters are good for many things, and I use most of the features on mine fairly consistently. The FTDI USB to UART bridge is a handy tool to have as it allows me to add a logic level UART port to my system that I would otherwise not have. The specific one that I use has the ability to switch between 5v logic and 3.3v logic via a jumper. UART, despite how old it is, it seems to be a protocol that is never going to

disappear from the environment of systems. And lastly there is the Digilent Analog Discovery, when I first got this device it was on a very strong recommendation from a trusted source. It was a very good purchase. This tool gives me 16 channels of real time digital I/O, 2 channels of oscilloscope, 2 channels of a function generator, a slight power supply, and an attempt at a multimeter. To be honest I purchased this for the digital channels features, but was pleasantly surprised to see that the majority of the other features on it were of good quality too. When working with digital systems and things aren't going right, it is easiest to see what is going wrong by looking at the 1's and 0's, and a logic analyzer does this most easily.

Design Rationale

As previously mentioned I divided this system into 3 smaller subsystems, and on each on I could have used something different than what I did in the end. To start things off the wireless system was something that I could have gone many different ways. In the past I have worked with WiFi modules, Bluetooth modules, and nrf24l01 modules. Each of these has their own set of pros and cons when it comes to communication wirelessly. What I decided to use was a 315MHz transmitter receiver pair that I had not used before, but I had an interest to see how they work. Part of the reason that I did not use any of the other three listed was because they all require setup and transmission routines, which adds to the complexity of using them. On top of this they are all fairly power hungry as they are all both transmitters and receivers, something I did not need as my system only needs to communicate in one direction, from the base to the remote. With the 315MHz unit not only was it less power hungry on the side that was on the remote, but it was also very easy to use. There is no protocol to use with it, just send it raw data and it transmits it. This did mean that I had to define my own simple packet protocol to use with it in order to make sure the data I was receiving on the remote was correct, but this was a minor concern and I still found it easier than using one of the protocols that the other three units required.

The next subsystem that I had to make a design choice with was the distance sensor. I ultimately use a XL-MaxSonar-WR unit that was donated to me earlier this year by the CIE department at Syracuse University. This unit make measuring distances very easy, as all you have to do is provided the unit with power that matches your logic level and it outputs the data via inverted UART on a pin. This is a slight issue as the inverted UART means that you have to invert the data before you process it, so to accomplish this I used a simple 1 transistor pull-up inverter circuit that I fed the UART signal through, and then read the output data into the BeagleBone Black, which was happy with this signal that came in. The units that the sensor outputs are centimeters, and as I was designing this product with the USA market in mind I had to do a conversion to inches, but this was done easily in code by simple multiplication.

I had many other distance sensors that I could choose from, but none of them were as easy to use as the XL-MaxSonar. Another option I had was to use a custom infrared system that was designed by Dr. Marcy a few years ago. This system is good in that for an IR based system it has a fairly high resolution, but it would require me to build one from parts, as the design is not new, but it is not kept assembled in the lab. Another issue with this system is that it relies on analog data, and a PWM on your processing system. Once the data is then read in as an analog value, the system then still needs to do some math with the input and output value to figure out the distance that it is reading. Plus due to variations in components, the system has to be calibrated for each time it gets built. All in all I would have probably gotten fairly accurate result with this system too, but it would have taken me a lot longer to use this as opposed to the XL-MaxSonar.

Another choice I had was to use an off the shelf IR sensor. There are many options out there when it comes to off the shelf choices, but the majority of them have a few issues that made it so I really couldn't even consider them. One of these issues is that the majority of them have a fairly limited range, on the order of a few centimeters to about 45 centimeters, which wouldn't work for me as my "stop point" for the car was decided at 36 inches. Another issue with these off the shelf units is that they only output a static value, either they see something or they do not. This would be okay if the distance on them was further and I was not going to display the current distance from the wall as I

did, but as I wanted to display the distance, I could not use them due to this design simplification.

The last subsystem that I had to make a decision on was the display. With the BeagleBone Black there are many options for displaying something, one could make use of the HDMI out on it and connect it to an external display, use one of the LCD capes of varying sizes made by CircuitCo, buy a display unit from the Adafruit collection, use a Hitachi based LCD with a parallel based data bus, or in my case you could use 7 segment displays. Earlier this year after doing some thinking on how to make the use of 7 segment displays easier, I came up with an idea of talking to them via a serial communication method. As it turns out this is not hard to do since the CD4543 will drive 1 7 segment display off of four input bits, and a 74HC595 is an eight bit shift register that when I use two CD4543 ICs can be driven by this shift register and allow for two digits to be displayed. I used this display party due to the availability of them and partly due to my understanding of how they work. Overall it would be able to display up to 99 inches on it, and this was a number that was out of range of my distance sensor anyways.

Results

Overall I would say that the results of my system were very positive. The distance sensor that I used was very fast at updating its value. One issue I had with it was that on the data sheet for the sensor it claims it can be used on a range from 0 inches to 195 inches, but after doing some testing of my own I found out that it was only able to be used in cases where it was between 11~94 inches. While this was disappointing to see, I was not terribly upset with this discovery, as for my project 94 inches was much further away than I needed to worry about, and 11 inches was much closer than I needed to worry about too.

With the Wireless solution that I used I was again very happy with the speed of it as it my packet protocol was only consisted of 4 characters, and this wireless solution was able to transmit at 9600bps, so it took not that much time to transmit my message each

time it needed to be transmitted. As the remote is the alarm part of the system, it needed to have a fast response rate, and this wireless solution allowed me to achieve this fast response rate that I needed.

For the display, it had one of the fastest communication protocols going to it, as it used SPI where the other two subsystems used UART. This allowed the display to update in near real time, and as fast enough to be comparable to human reaction times. Due to the display only being two digits big I designed the code for my system to stop displaying any value larger than 99 inches, and since my sensor only seemed to work up to 94 inches this was a good choice as when the distance sensor goes out of range it starts outputting its largest possible distance, 195 inches, so turning off the display was a good choice. The flipside of this is that the sensor can only see things that are at least 11 inches away, and when it is less than 11 inches away, it just outputs 11 inches, so similar to my too large of a value issue, I decided that any values reported less than 12 inches would result in the display getting turned off as well.

With combining the systems together I used a BeagleBone Black running Debian 7.9 with the Linux 3.8.x kernel on it. This allows me to use the libraries that I have written to communicate with the SPI and UART in C++. Since I had tested all these subsystems independently of each other prior, but not with the BeagleBone Black, I made some quick test to see how it was communicating with these devices and the BeagleBone Black. Once I was happy with the communications I wrote some C++ code that samples the sensor, converts the value, checks the value for certain ranges, and based on the range it falls into does a certain task, so if it is less than 12 it ignores it and blanks the display. If it is between 12 and 99 it will display the value on the display, if the value is between 12 and 36 it will sound the alarm on the remote and only turn the alarm off if the value goes above 40. The last range it checks is if the sensor is above 99, if this is the case then it does the same as the first case and turns off the display.

Again overall I am very happy with the results that my system was able to show. Both the display and the remote had a fast enough response rate that human reaction time was comfortable with responding in time. As these results were very positive, I would call this system a success and the project as one too.

Future Work

If I was to spend more time on this project I would be interested in adding an iot (internet of things) component to it. This would require that I add a WiFi module to the system, and as the BeagleBone Black runs Debian and has a USB deice port, that wouldn't be too difficult. The real challenge here would be what data to report and how to report it. Some ideas that come to my mind are: tweeting when you park and how close you parked, sending the data to a spreadsheet program to plot how accurately you were at parking exactly 36 inches away from the wall, or even a simple game where you gain points based on how accurate your parking was that day.

Another area that I might give some time to if I was to revisit this project would be the wireless system, I chose it because it was quick, easy, dirty, and cheap. But all these things also provided their own issues at points, Transmitting the danger message required me to send it many times to make sure that the remote got them, as did transmitting the clear message. This resulted in cases where the messages might not go through as quickly as I would have liked, but again, the messages got through, and I was satisfied with that.

Conclusion

In total I am very happy with the results of my project. All of the task that I had set out to accomplish I managed to do. The system is able to detect an object getting closer to it and display the value on the screen, once the item is within a certain range my system is able to talk to a remote and alert the user via the remote that they are at the appropriate distance and to stop backing up. While the remote unit did not really have any housing designed for it, I was pleased with the final appearance of the base unit. My concept of designing the system first as a series of sub systems to then integrate together was an efficient one as when it came time to fully construct the system I was able to do so as I understood all the components and how they operated. This project

allowed me to explore the BeagleBone Black as a product as opposed to an educational tool, which was how I was most familiar with it previously.

References

Arduino. Arduino - Serial. 2016. April 2016 <https://www.arduino.cc/en/Reference/Serial >.
Geramian, Philip D. Beaglebone/LIBS. 30 March 2016. April 2016 <https://github.com/saterblader/Beaglebone/tree/master/LIBS >.
Marlin P. Jones & Assoc, Inc. MPJA. 28 September 2014. April 2016 <http://www.mpja.com/download/31960MPSch.pdf >.
MaxBotix. MaxBotix. 2015. April 2016 <http://www.maxbotix.com/documents/XL-MaxSonar-WR_Datasheet.pdf >.
Molloy, Derek. Exploring BeagleBone: Tools and Techniques for Building with Embedded Linux. Dublin: Wiley, 2014.
NXP. 74HC595. 25 February 2016. April 2016 <https://www.nxp.com/documents/data_sheet/74HC_HCT595.pdf >.
Texas Instruments. CMOS BCD-to-Seven-Segment. March 2002. April 2016 <http://www.jameco.com/Jameco/Products/ProdDS/13696.pdf >.
Woodyard, Chris. "NHTSA to require backup cameras on all vehicles." USA Tody 1 April 2014.