TRIBHUVAN UNIVERSITY

Institute of Engineering

Department of Electronics and Computer Engineering

"GPS Based Trigger"

By
Natasha Shrestha (059BEX417)
Lalita Bhetwal (059BEX413)
Pooja Chhetri (059BEX425)

December 2006

2. Abstract:

Since time immemorial, human beings have been trying to invent new things for their comfort and ease in leading their life. People are prone to newer ideas and technologies. Location-based-alarm system is an addition to these developing technologies. This system enables the user to find his/her area of interest e.g. Hotels, restaurants, hospitals, banks, etc. This system is aimed at making the life of people easier and more systematic.

Positioning is a very necessary factor in our life. Our project enhances the services offered by a GPS receiver. This project removes the GPS from the category of just a handheld “help-me-find-my-way-home” and keeps it in more useful tools category. A GPS receiver is aimed at giving the precise position in latitude and longitude of a place. We use this feature of GPS receiver to find the area of our interest. For example we may need to fill fuel in our tank and may not know where the nearest petrol pump is, and may just pass by one without knowing that it was there. At such times we may be benefited by a location-triggered-alarm system, which gives an alarm when our desired place which may be hotel, restaurants, banks, petrol pumps, etc. is near.

3. INTRODUCTION

Our ancestors had to go to pretty extreme measures to keep from getting lost. They erected monumental landmarks, laboriously drafted detailed maps and learned to read the stars in the night sky for navigation. Things are much, much easier today. For less than $100, one can get a pocket-sized gadget that will tell you exactly where you are on Earth at any moment.

Alarms are often used as reminders of tasks that need to be performed at preset times. Handheld devices have the advantage that the users can carry such devices with them wherever they travel, so that the users are able to hear or see the alarm when the alarm is activated.

However, basing an alarm only on time reduces flexibility to a user. An alarm may pertain to a task to be performed by a user at a given location. However, when the alarm is activated at the preset time, the user may be far away from that location. If a user is unable to perform the task close to a time at which the alarm is activated, the user may forget to return to the task at a later point in time, which defeats the purpose of such an alarm.

Many things that a person has to do are associated with particular places. For example, one mails a letter at a post office or a mailbox, buys groceries at a local grocery store, and checks the condition of furnace filters at home. There is presently no easy way for a person to be reminded of something (e.g., a to-do item) when he or she arrives at a corresponding location. Hence, a person is usually required to keep the association of the action and the corresponding location at the forefront of their mind, and thus subject to be forgotten.

The main aim of our project is to construct a location based reminder which is equipped with a global positioning system (GPS) receiver and a microcontroller and is programmable by the user to alert the user when he or she arrives with the device at a predetermined location.

The reminder application can give the user relevant information at a given location like 'your car needs fuel and you are near a petrol pump now' or 'you are hungry now and there is a good restaurant nearby'. But first the user has to choose or enter the mode that he wants to be in, like -hungry mode, refuelling mode, sick mode, banking mode, etc. Then the alarm will remind you if you are near a restaurant and the hungry mode is on, or when you are near a petrol pump and you are in refuelling mode, etc.
We will need to create a location sensitive database which is already programmed in the microcontroller. As the user drives on, the microcontroller checks the location of the user sent by the GPS receiver with the location in the system's database for the specific mode and if a match is found, the user is reminded by an alarm system. If the user wants to find yet another place, then he doesn't remove the mode and he will again be reminded when a match between the user's location and the database takes place.

OVERALL BLOCK DIAGRAM:

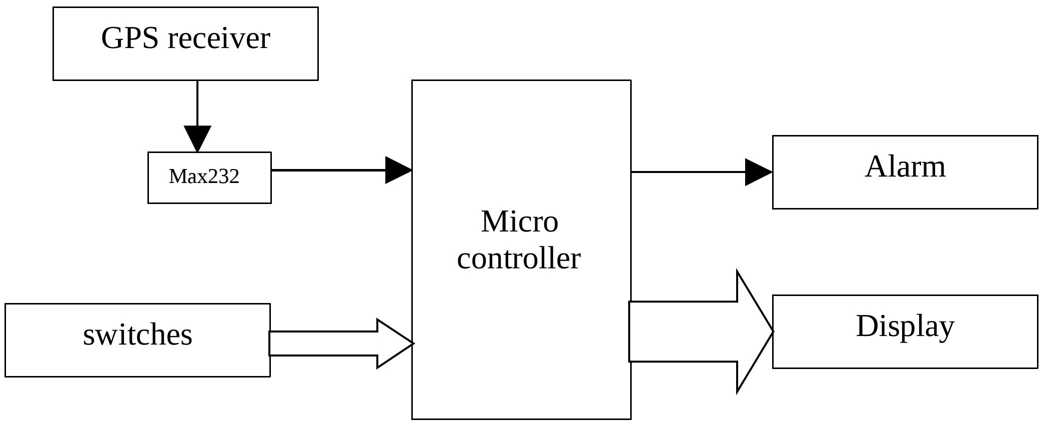


Fig. Overall Block Diagram

3.1 <u>GPS Receiver :</u>

The Global Positioning System (GPS) is actually a constellation of 27 Earth-orbiting satellites (24 in operation and three extras in case one fails). The U.S. military developed and implemented this satellite network as a military navigation system, but soon opened it up to everybody else. Each of these solar-powered satellites circles the globe making two complete rotations every day. The orbits are arranged so that at any time, anywhere on Earth, there are at least four satellites "visible" in the sky. A GPS receiver's job is to locate four or more of these satellites, figure out the distance to each, and use this information to deduce its own location. This operation is based on a simple mathematical principle called trilateration.

The GPS Receiver we are using is of Garmin, which support a computer interface. In addition Garmin supports differential gps input signals. The modes supported by Garmin receivers are NMEA and others, which are not to our concern. The hardware interface for Garmin units meets the NMEA requirements and is sufficient to drive 3 NMEA loads. It is also compatible with most computer serial ports using RS232 protocols. There is only a data in and data out line with ground. There are are no handshake lines nor should we attempt to set up a software handshake.

To work with the Garmin GPS, first we should be sure that we have selected the correct mode and baud rate for the program or unit we are trying to interface with. This is the main problem with interface failures. For all moving map programs we likely need NMEA mode with the baud rate set to 4800. In all cases, we should set the data width to 8, no parity, and 1 stop bit and also the correct com port, while interfacing it with the computer serial port.

3.2 GPS Protocol:

The National Marine Electronics Association (**NMEA**) has developed a specification that defines the interface between various marine electronic equipment. The standard permits marine electronics to send information to computers and to other marine equipment. GPS receiver communication is defined within this specification. Most computer programs that provide real time position information understand and expect data to be in NMEA format. This data includes the complete PVT (position, velocity, time) solution computed by the GPS receiver. The idea of NMEA is to send a line of data called a sentence that is totally self contained and independent from other sentences. There are standard sentences for each device category and there is also the ability to define proprietary sentences for use by the individual company. All of the standard sentences have a two letter prefix that defines the device that uses that sentence type. For gps receivers the prefix is GP. This is followed by a three letter sequence that defines the sentence contents. In addition NMEA permits hardware manufactures to define their own proprietary sentences for whatever purpose they see fit. All proprietary sentences begin with the letter P and are followed with a letter that identifies the manufacturer controlling that sentence. For Garmin this would be a G.

Each sentence begins with a '$' and ends with a carriage return/line feed sequence. The data is contained within this single line with data items separated by a comma. The data itself is just ascii text and may extend over multiple sentences in certain specialized instances but is normally fully contained in one variable length sentence. An example sentence might look like:

```
 $GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*42

With an interpretation as follows:

     GGA - Global Positioning System Fix Data
     123519       Fix taken at 12:35:19 UTC
     4807.038,N   Latitude 48 deg 07.038' N
     01131.000,E  Longitude 11 deg 31.000' E
     1            Fix quality: 0 = invalid
                               1 = GPS fix
                               2 = DGPS fix
     08           Number of satellites being tracked
     0.9          Horizontal dilution of position
     545.4,M      Altitude, Meters, above mean sea level
```

```
46.9,M       Height of geoid (mean sea level) above WGS84
                     ellipsoid
(empty field) time in seconds since last DGPS update
(empty field) DGPS station ID number
*42          the checksum data, always begins with *
```

Each Data type would have its own interpretation that is defined in the NMEA standard. This particular sentence provides essential fix data. Other sentences may repeat some of the same information but will also supply new data. Whatever is reading the data can watch for the data sentence that it is interested in and simply ignore whatever sentences that is doesn't care about. In the NMEA standard there are no commands to indicate that the gps should do something different. Instead each receiver just sends all of the data and expects much of it to be ignored. On NMEA input the receiver stores information based on interpreting the sentence itself. While some Garmin receivers accept NMEA input this can only be used to update a waypoint or similar task and not to send a command to the unit. There is no way to indicate whether the sentence is being read correctly or to re-send some data we didn't get. Instead the receiving unit just checks the checksum and ignores the data if the checksum isn't correct figuring it will be sent again sometime later. No error can be generated to the remote system.

The NMEA standard has been around for many years and has undergone several revisions. The protocol has changed and the number and types of sentences may be different depending on the revision. All Garmin receivers understand the latest standard which is called: 0183 version 2.0. This standard dictates a transfer rate of 4800 baud.

3.3 Serial Communication:

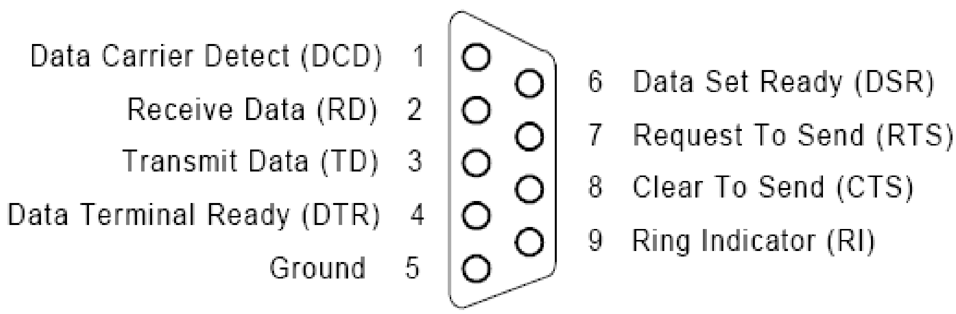
Serial communications is the process of sending data one bit at one time, sequentially, over a communications channel or computer bus. This is in contrast to parallel communications, where all the bits of each symbol are sent together. Serial communications is used for all long-haul communications and most computer networks, where the cost of cable and synchronization difficulties make parallel communications impractical. Serial computer buses are becoming more common as improved technology enables them to transfer data at higher speeds.

There are two types of serial communication: **Asynchronous and Synchronous.**

Most PC serial devices such as mice, keyboards and modems are asynchronous. Asynchronous communication requires nothing more than a transmitter, a receiver and a wire. It is thus the simplest of serial communication protocols, and the least expensive to implement. As the name implies, asynchronous communication is performed between two (or more) devices which operate on independent clocks. Therefore, even if the two clocks agree for a time, there is no guarantee that they will continue to agree over extended periods. To combat this timing problem, asynchronous communication requires additional bits to be added around actual data in order to maintain signal integrity. Asynchronously transmitted data is preceded with a start bit which indicates to the receiver that a word (a chunk of data broken up into individual bits) is about to begin. To avoid confusion with other bits, the start bit is twice the size of any other bit in the transmission. The end of a word is followed by a stop bit, which tells the receiver that the word has come to an end, that it should begin looking for the next start bit, and that any bits it receives before getting the start bit should be ignored. To ensure data integrity, a parity bit is often added between the last bit of data and the stop bit. The parity bit makes sure that the data received is composed of the same number of bits in the same order in which they were sent.

Most PC interfaces for serial data exchanges follow the RS-232c standard of the EIA (Electronic Industry Association). The standard defines the mechanical, electrical, and logical interface between data terminal equipment (DTE) and a data carrier equipment (DCE). The DTE is usually formed by a computer and the DCE by a modem. The IBM defines a 9-pin connector for its serial interface, where two of the usually present RS-232C signals are missing.

On the 9-pin connector, the protective ground and the signal for the data signal rate are missing but the remaining nine signals are sufficient for a serial asynchronous data exchange between a DTE and a DTE in accordance with the TS-232C standard. The pins 3/2 and 2/3 transfers the data signals; the rest of the connections are intended for the control signals. Its structure is shown below:



PC Computer RS-232 Connector Pin Assignment, 9 Pin Connector

For controlling the data transfer between DTE and DCE, the five control signals-RST,CST, DCD, DSR and DTR are decisive. But we have only used two pins, the data-in and the ground. Other pins are not to our concern. We have not made the use of handshaking pins: RTS and CTS.

3.4 Universal asynchronous receiver/transmitter (UART):

A **universal asynchronous receiver/transmitter** is a type of "asynchronous receiver/transmitter", a piece of computer hardware that translates data between parallel and serial interfaces. Used for serial data telecommunication, a UART converts bytes of data to and from asynchronous start-stop bit streams represented as binary electrical impulses. UARTs are commonly used in conjunction with other communication standards such as EIA RS-232.A UART is usually an individual (or part of an) integrated circuit used for serial communications over a computer or peripheral device serial port.

The UART usually does not directly generate or receive the external signalling levels (such as voltages on wires) that are used between different equipment. Typically, an interface is used to convert the logic level signals of the UART to the external signalling levels. "Signalling levels" is a very broad term encompassing all the various possible schemes to convey a level from one place to another. Voltage is by far the most common kind of signalling used. Examples of standards for voltage signalling are RS-232, RS-422 and RS-485 from the EIA. As of 2006, UARTs are commonly used with RS-232 for embedded systems communications. It is useful to communicate between microcontrollers and also with PCs. Many chips provide UART functionality in silicon, and low-cost chips exist to convert UART to RS-232 signals (for example, Maxim MAX232).

3.5 Hardware Used:

3.5.1 Microcontroller:

We are using AT89C52 as our microcontroller. It is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of Flash Programmable and Erasable Read Only Memory (PEROM). The device is manufactured using Atmel’s high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C52 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications. Its pin layout is as follows:

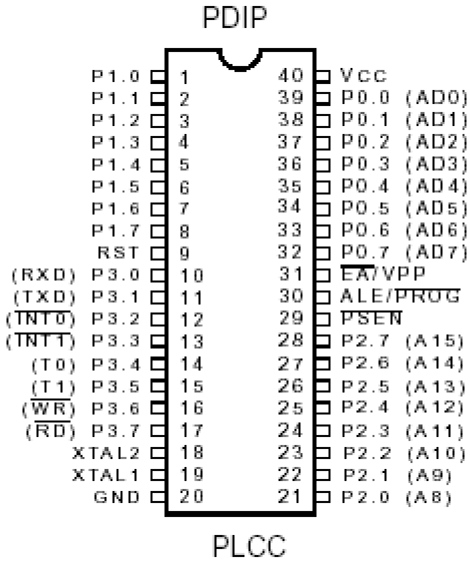


Fig: Pin Layout

3.5.2 <u>MAX 232</u>:

The MAX232 device is a dual driver/receiver that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V and a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The MAX232 is characterized for operation from 0°C to 70°C.

MAX 232 is a RS232C Level Converter. A standard serial interfacing for PC, RS232C, requires negative logic, i.e., logic '1' is -3V to -12V and logic '0' is +3V to +12V. To convert a TTL logic, say, TxD and RxD pins of the uC chips, thus we need a converter chip. A MAX232 chip has long been using in many uC boards. It provides 2-channel RS232C port and requires external 10uF capacitors. A DS275, however, doesn't need external capacitor and is smaller. Either circuit can be used without any problems.

The top view of MAX 232 is as shown:

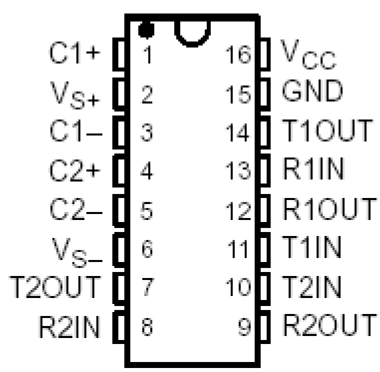
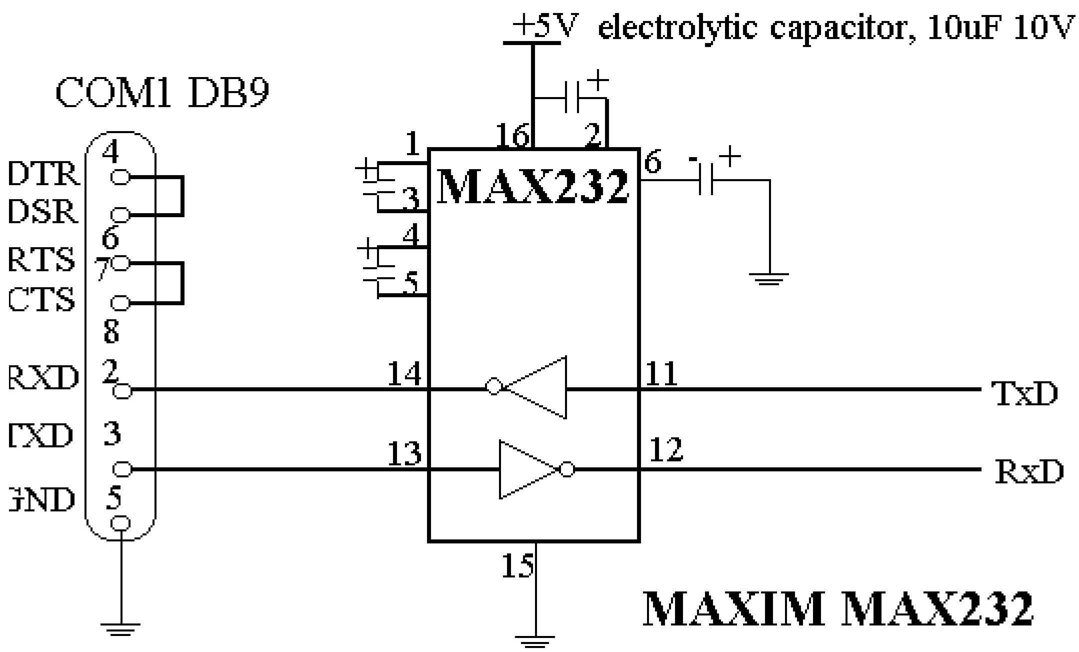


Fig:Top View

The pin configuration its connection with a RS-232 is as shown:



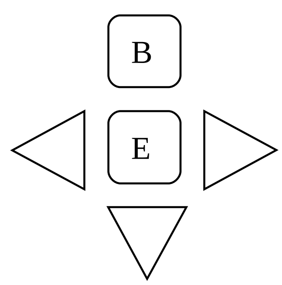
3.5.3 <u>Liquid Crystal Display (LCD)</u>:

The LCD we have used in our project is the JHD162A which is a 16 CHAR x 2ROW series display. The LCD and its pin configuration are as follows:

Pins	Description
1	Ground
2	Vcc
3	Contrast Voltage
4	"R/S" _Instruction/Register Select
5	"R/W" _Read/Write LCD Registers
6	"E" Clock
7 - 14	Data I/O Pins

3.5.4 Switches:

As our system is based on user interaction, we need the switches for providing inputs to our system. We have used altogether five switches which are shown below:



where '**B**' stands for Back and '**E**' stands for Enter. Use of others are clear from the figure itself. The swiches are used for moving to different screens in the LCD and also editing data and locations.

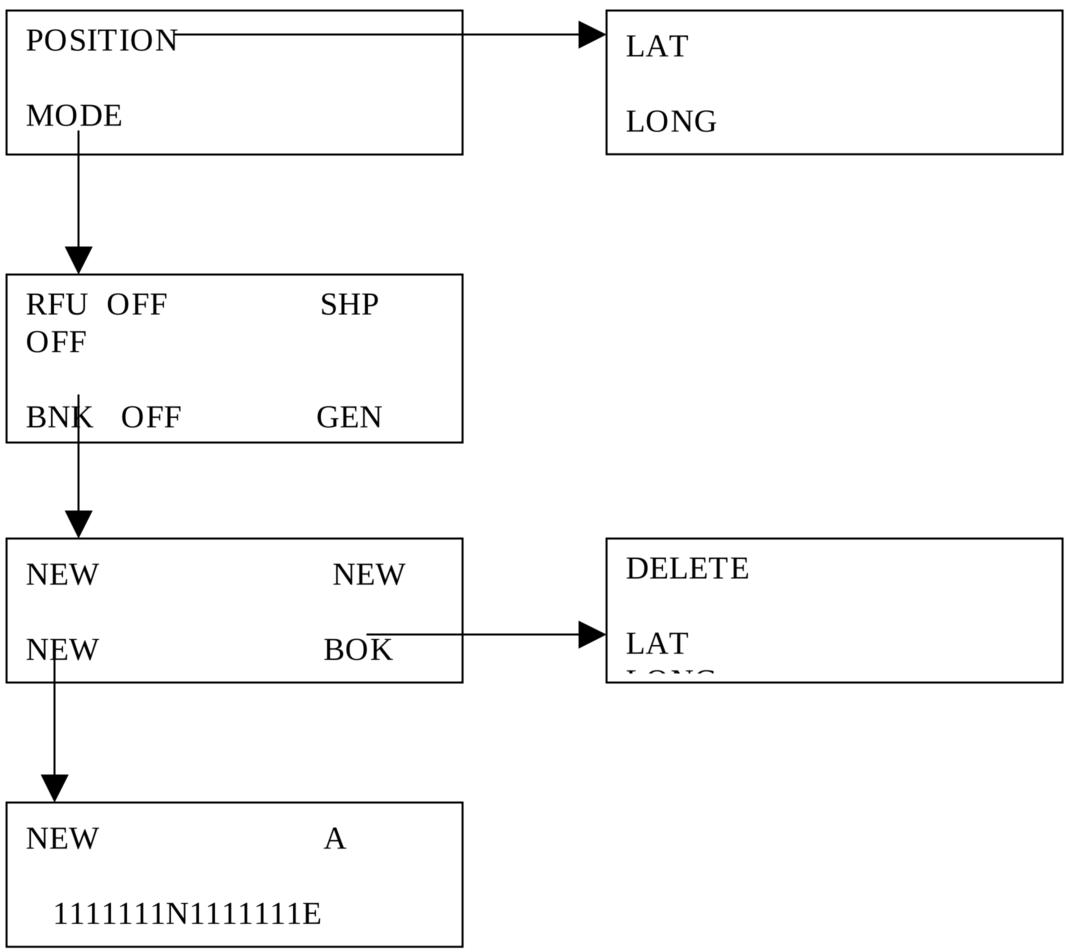
4. PROGRESS:

Concerned to our Project progress, so far we have completed
Hardware portion:

- ❖ Interfacing GPS Receiver with microcontroller
- ❖ Interfacing LCD and switches with microcontroller

Software portion:

In this, we have accomplished four user interactive screens as shown below:



5. WORK REMAINING:

In the hardware portion the remaining work is the connection of alarm and in the software portion the timer control part is left, which includes retrieving data from the GPS Receiver and using it to trigger the alarm. Besides, further programming for the user interface is also to be accomplished.

6. CHALLENGES FACED:

The main difficulty we have faced is the interface with the GPS Receiver. Due to the GPS Receiver's inability to retrieve the data inside buildings, we faced a lot of problems during our working period.

As our project deals with user interaction, the program consumes more memory so we are facing problem due to limited memory of the microcontroller that we have been using. Because of the unavailability of the GPS Receiver in our department, we faced a problem in obtaining it and we have not been able to use it continuously throughout our project.

7. LIMITATIONS:

As we have not used the EEPROM, the stored data in the microcontroller gets lost as soon as the power goes off. Due to the limited memory, we can store only 16 way points.

8. CONCLUSION:

Our project aims at easing the life of people by the help of this location based alarm in reminding a to-do item when he or she arrives at a corresponding location. Till now, we have only time based alarms which prove insufficient at times when locations are of main concern. A person is usually required to keep the association of the action and the corresponding location at the forefront of their mind, and thus subject to be forgotten. Our project hopes to accomplish the task of efficiently reminding us of the things that we have to do according to the location we are in.