Smart Attendance System

1Mahantesh S 2 Mahesh H 3Ramchandra K 4Shivanand K 5Rohini H

BVBCET, Vidyanagar, Hubli-580031

[1mahantesh.salimath130@gmail.com](mailto:1mahantesh.salimath130@gmail.com), [2mahesh.hegde129@gmail.com](mailto:2mahesh.hegde129@gmail.com), [3ramchandraak@gmail.com](mailto:3ramchandraak@gmail.com), 4[shivanandk58@gmail.com](mailto:shivanandk58@gmail.com), [5rohini\_sh@bvb.edu](mailto:5rohini_sh@bvb.edu)

**ABSTRACT:**

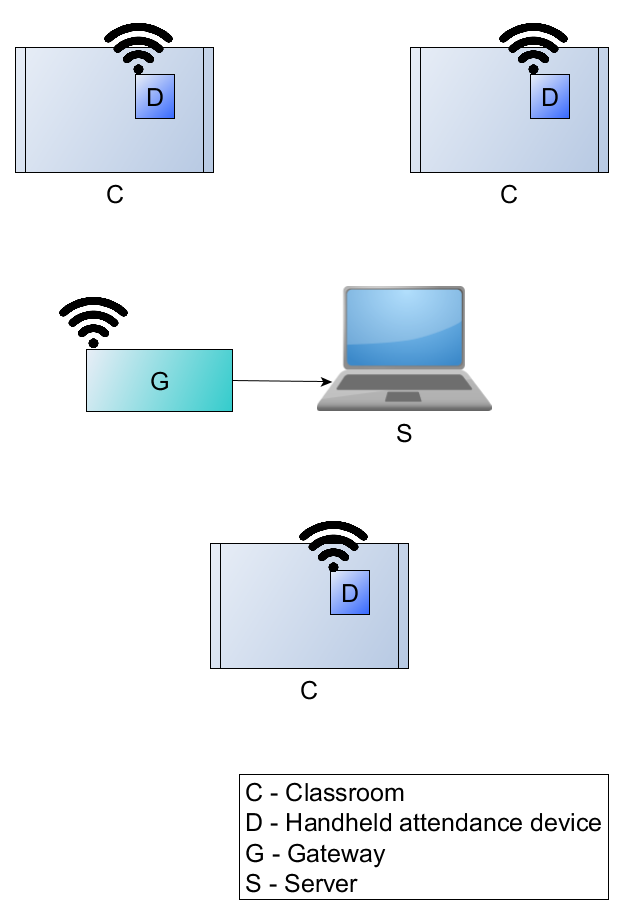
Internet has been entering into every aspect of today's life. Every device is becoming smart by staying connected to the internet. Attendance monitoring is one area where the usage of smart devices is limited. In today's busy life, parents are not able to track the day- to-day activities of their children, they are not even aware of whether their children are attending their classes regularly. Attendance plays a very important role in modern education system. To ensure that students attend their classes on a regular basis, Educational institutions are taking strict measures. But the existing systems have flaws in them, they are not entirely accurate, for example, the Professor may fail to recognize any proxy attendances and it is also an overhead on the Professor to update the attendance on website. Taking attendance in class as well as updating it onto the website is time consuming and extra burden. If the attendance is not updated on time, the parents are unaware of their ward's status. This problem has been overlooked for a long time. The following paper aims to provide a novel solution to the problem.

**Keywords:** Biometric attendance, Mobile attendance monitoring

1. **INTRODUCTION**

A variety of attendance monitoring systems are employed in schools, colleges and organizations. In schools and colleges, the attendance of each student is taken manually by the teacher. The teacher notes down roll numbers of students present for that class and updates it on to the website later. On an average 5-10 minutes are spent in taking attendance of each class. The above steps are to be carried out by each teacher for every class he/she engages. The attendance data of students is not updated onto the website everyday but, once in a week or in a fortnight. In various organizations, automated attendance systems are used which are based on electronic tags, barcode badges, magnetic stripe cards, biometrics etc. These systems are static and are mount at specific places. Every individual has to approach the module to cast his attendance. Thus, these systems cannot be employed in schools and colleges as they involve taking attendance of a large number of students on an hourly basis. We aim to address the issues in existing attendance monitoring system by designing Smart Attendance System. It is a portable biometrics based attendance system which updates the attendance directly onto the website without teacher0s intervention.

1. **METHODOLOGY**



**Figure 1: Block diagram for Smart Attendance System**

A typical use case of Smart Attendance System in a school/college environment is shown in figure 1. Handheld devices are used in different classrooms to record the attendance and a server updates the same into a database.

**Block description**:

Smart Attendance System comprises of:

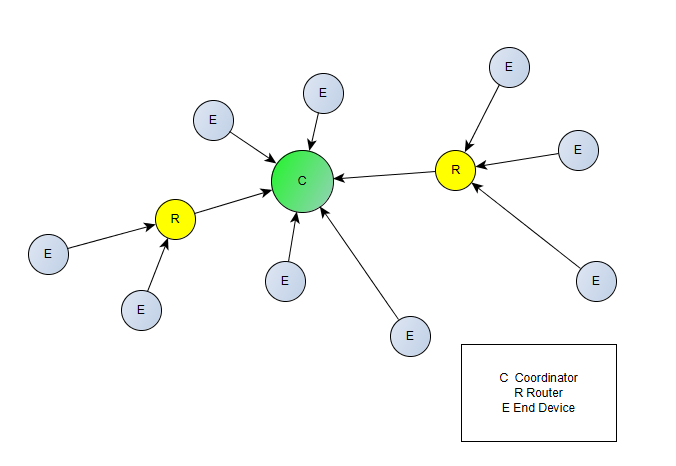
1. Multiple Handheld devices
2. Gateway device
3. Server

Fingerprint module:

Each handheld device is made up of a fingerprint module, a microcontroller and a Zigbee transceiver. The fingerprint module is a biometric device used for student identification. It comes with a storage to save user's fingerprints and an on-board processor to identify them later on. Depending on the size of class, modules of different capacity can be used. The fingerprint module attaches a unique ID for each fingerprint stored and on identification, returns the same ID. The fingerprint module sends the ID of user on a serial interface, which is interpreted by the microcontroller.

Zigbee:

For the wireless communication between handheld device and the server, ZigBee-enabled devices are used. Xbee is one such device which implements the ZigBee standard. In a ZigBee network, the nodes can be classified as coordinator, router, and end device. In brief, a coordinator is responsible for forming the ZigBee sensor network, the router is responsible to route data between different end nodes in the network, end devices cannot route data they are either connected to a router or the coordinator and are designed in a way to consume low power.



Communication using XBee is possible in two modes, AT (Transparent) mode or API (Application programming Interface) mode. AT mode is limited to fixed point-to-point communication between two XBee devices. In API mode it is possible to send and receive data from any XBee device in the network. Additionally, other information can also be exchanged such as knowing the status of remote IO and ADC and also control them, feedback of the packet reception etc.

In this project, the Xbee present at the server end is configured as the coordinator. It acts as a gateway device between the sensor network and the remote server PC, it is configured in API mode as it needs to know the address of the remote XBee which sent the data to it. The Xbee in the handheld device is configured as an end device with AT mode as the destination address for them is fixed with the address of the coordinator. The frame format received at the coordinator is explained here: https://docs.digi.com/display/XBeeZigBeeMeshKit/Frame+structure The Xbee transceiver is a radio frequency transceiver operating at 2.4GHz frequency. Zigbee can operate at distances ranging 100m and hence is suitable for the application.

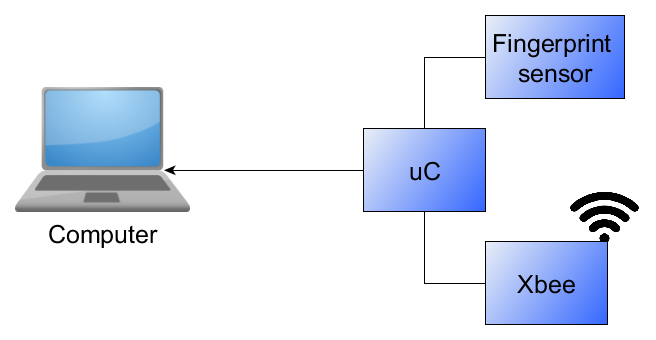
Microcontroller

Both fingerprint module and Zigbee device are interfaced to a microcontroller. The microcontroller receives information from the fingerprint module, interprets it and then intelligently transmits the required data to gateway device via Xbee. ATMega328P was used as the controller whereas any microcontroller which supports 2 UARTs can be used for this application.

Server:

The server is serially connected to the gateway device (Xbee coordinator) which provides the data received from various handheld devices. The server decodes it, and updates the central attendance database.

1. **IMPLEMENTATION**
   1. **Fingerprint Enrollment:**

****  
Figure 3.0 Enrollment setup

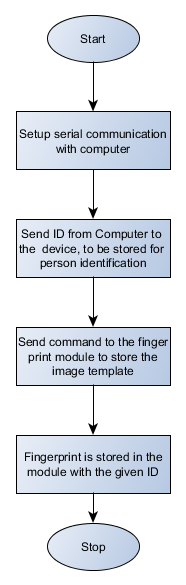
The flowchart in figure 3.2 shows the enrollment procedure of students0 fingerprints. The fingerprint module is connected to the microcontroller and they communicate with each other using serial communication at a baud rate of 57600. The controller is also initialized for serial communication with a computer at 9600 baud. A unique identification number (Roll number) is sent via computer to the controller serially. This identifier is used by the controller to command the fingerprint module to store the thumb impression template at the ID specified. Thus each fingerprint stored is associated with a corresponding ID. For the enrollment of each fingerprint, two images are taken and the information from both the images is used to generate a template which is then stored in the module. This template is used for search operations and whenever a match is found, the corresponding ID is sent serially to the microcontroller.  


Figure 3.1 Enrollment flow

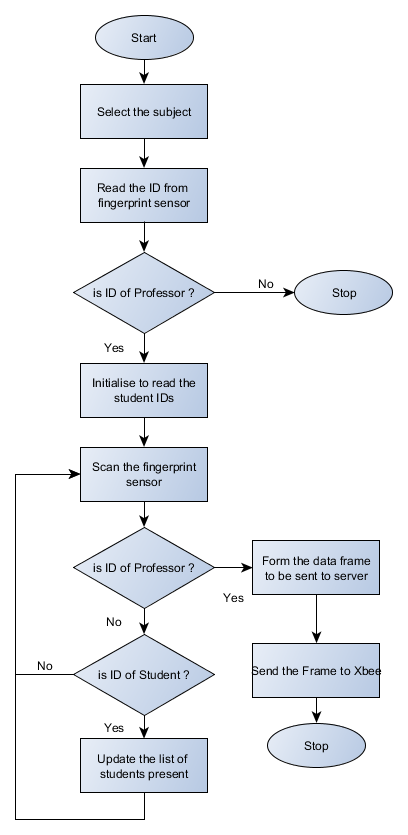
Figure 3.1 explains the series of actions performed by the controller during the enrollment of a new fingerprint.

* 1. Updating attendance

The first step in taking the attendance is Subject Selection. The professor selects the relevant subject using a button interface. As he presses the button, subjects pertinent to the class are iterated and are displayed on the LCD. He then confirms it by asserting his fingerprint on the sensor. The device is then passed onto the students so that they can register their attendance. After all the students have recorded their attendance, the professor closes the session by asserting his fingerprint again. Now, the device frames the attendance data and sends it to the server. The figure 3.3 depicts the flow explained above.

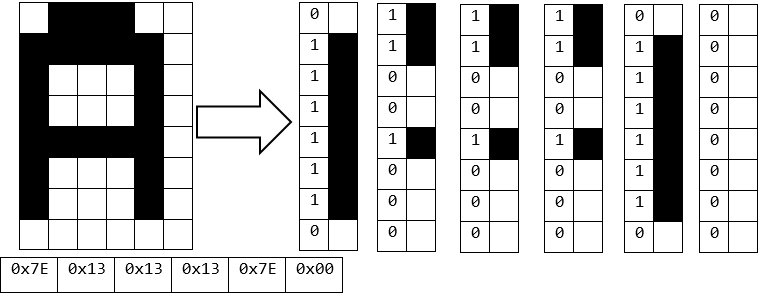
1. Frame format:
2. Minimizing payload:
3. Timeout handling:

Communication between handheld device and gateway



**3.2 Character**

To display characters on the screen, a character array database is formed. This database consists of the binary data which gives the pixel arrangement for each character. The database consists of characters of different sizes viz., 4x6, 8x6, 9x12, 11x16. An example of implementation of displaying of a character is shown in the Figure 3. Similar implementations have been observed in [3].



**Figure 3: Character with 8x6 Resolution**

Each character is an arrangement of pixels. The array specifies which pixels are to be lit in order to form the character. The binary value `1' specifies the pixel corresponding to a character and `0' specifies the pixel corresponding to the background.

An API - print\_string\_lcd is implemented for displaying of characters. The function has a total of eight arguments.

The first two arguments specify the row and column of the screen where the first character has to be printed. The number of rows and columns that the screen can accommodate is calculated by Equation 3.1 and Equation 3.2 respectively.

Max.NoofColumns=(No.pixelsinRow)/(Characterwidth)

3.1

Max.NoofRowss=(No.pixelsinColumns)/(CharacterHeight)

3.2

The above two values serve as the upper limits for displaying of the characters on the screen. The limits are unique for every size of the character. The value in equation helps to decide when to start a new line by detecting the overflow in the upper line.

The next argument is used to specify the color of the character to be displayed on the screen. The color is in 16 BPP format. Similarly the argument `back' is used to specify the background color of the character. This helps to dwell the characters with the previous background color.

The next two arguments specify the size of the character and whether underlining of the character has to be done or not. The size argument takes SIZE\_4X6, SIZE\_6X8 and SIZE\_11X16 as inputs which are pre-defined in the programmed. The underline argument takes U\_LINE and N\_ULINE as the arguments. U\_LINE specifies that the character have to be underlined and N\_ULINE specifies no underline.

The last argument takes character pointer as the input which specifies the string that is to be printed on the screen.

The function print\_string\_lcd internally calculates the max column and max row count depending on the character size specified. Then the each character is printed individually on the screen. Each character is then fetched from the character pixel array. The character is then plotted pixel by pixel with the help of lcd\_plot routine.

**3.3 Message Box**

The basic entity which is used to create the message box is a rectangle which is in turn done with the help of lines. A line is drawn on the LCD screen by knowing the end points. Assuming (x1,y1) and (x2,y2) are the two user specified points the line is drawn using the Equation 3.3.

3.3

The pixels which fall between these end points are illuminated on the LCD screen with the user specified color. The rectangle is formed using this method to draw lines. The title field and the message field are partitioned by drawing a line at a specific distance from the top of the message box. The colors of the two areas are user defined and can be colored with two different colors. The shadowing effect to the message box is given by illuminating the pixels which are near to the outer edges of the message box.

Centering for the text is achieved by knowing the length of the text which we need to display and the size of the message box in terms of its length. This again depends on the size of the character which the user is displaying as the change in the size of the text changes to number of pixels required to form the letters of the text.

Using the length of the message box, the center of message box is calculated. The center of the message box is aligned with the center of the text which is to be displayed and thus centering is achieved.

An API - am1810\_msgbox is implemented for displaying message box on the screen. The function takes three arguments as inputs. The first argument takes title of the message box as input in the form of character pointer. The second argument takes the message field as the input in the form of character pointer. The last field defines whether the message box should wait for user prompt or not.

The am1810\_msgbox () routine internally draws lines to form the rectangular shape. The rectangular shapes are filled with color and the `title' and `message' areas are differentiated by drawing a line and filling with a different color. Similarly `OK' and `CANCEL' fields are provided at the bottom of the message box depending on which the further action is processed.

**3.4 Serial Communication**

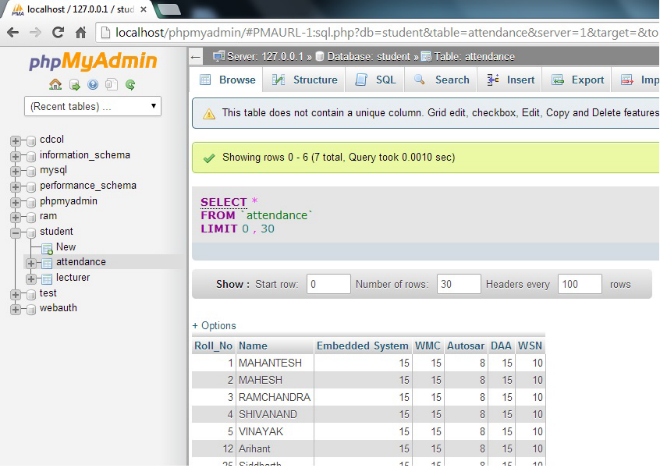
Serial communication is required to establish communication between the boards to other devices. In this context the communication is with a PC. The baud rate is set as 9600 with 8 bit data, 1 start bit, 1 stop bit and no parity mode. With the help of serial communication, user can enter text on the LCD screen. Serial communication here is used in such a manner that the whole project can be demonstrated.

When the serial communication is established a message box is displayed on the screen showing that communication has been established. The next screen will be a text pad where user can edit text on the LCD screen via the keyboard connected to the PC.

Keywords are provided to the user to access other features viz., EXIT - used to exit the text pad. IMAGE> - used to display a message box giving the options of different images. The message box requires user prompt for the further action from this point. Upon user input, a particular image will be displayed on the screen for five seconds.

Thus the User is able to interact with the hardware using serial communication and all the APIs are implemented to provide full access to User at each point of interaction with the hardware.

**4 RESULTS**

****



**4.1 Display of Images**

The Figure 4 shows the display of User defined images. For the display of an image, the input data to the LCD controller is the RGB pixel array, which is extracted from input image.



**Figure 4: Display of user defined image**

The data flow involves pixel data being put in a frame buffer, which stores pixel data sufficient to fill the LCD screen. The data then flows to the raster controller, which is responsible for the pixels being illuminated and the image being displayed.

The built-in DMA engine constantly transfers the pixel data stored in the frame buffers to the input FIFO, from where the raster controller relays the data to the output pins.

**4.2 Display of Characters**

The Figure 5 shows the display of characters from User input. The characters are displayed on the screen according to the user inputs by mapping the already present database of each character in the form of arrays of pixel values to the input character provided by the user.



**Figure 5: Display of character from user input**

The database for each character has been developed depending upon what pixels are to be illuminated in the given area and these are stored in the form of array values which are illuminated on the LCD screen on a 2-D space.

**4.3 Display of Message Boxes**

The Figure 6 shows the display of message box for User information. A message box is created using simple geometric shapes like lines. A rectangle is first created using equation of line, which gives the shape of a box. Then, the rectangle is divided into fields such as the `title', `message' and `prompt' field.



**Figure 6: Display of message box**

The `title' field contains the title of the message box, as the name suggests.

The `message' field contains the message to be printed for the user and finally the below boxes depict the `prompt' options for the User, depending on whose input, the further process follows.

**5 CONCLUSION AND FUTURE WORK**

The Non-OS based LCD driver is a piece of software which acts as a tool to interact with the hardware . This paper designs a software that illuminates the LCD screen, and the various ways in which the LCD screen is illuminated involves displaying of sample images on the LCD screen, displaying dialogue boxes on the LCD screen, creating a database of various character fonts, displaying the same and creating shapes of various sizes and colors.

Conversion of image into its corresponding pixel array data structure using Image Convert software. Creating an API that helps to display any stored image onto the LCD Screen.

Storing each character of different resolution in separate array data structures in the form of their corresponding pixel values.

Creating an API for Character display that provides an User option for selecting particular Character, its Resolution, Text color, Background color and Underlining.

Creating simple rectangular boxes using equation of line and improvising it with shadow border effects. Creating an API for Message Box, providing user input for Title, Message and other Prompt Fields, depending on which the further action is taken.

**FUTURE WORK**

This LCD Driver is implemented on a NON-OS platform keeping in mind the unnecessary overhead of an Operating System to drive an LCD. Hence, only the routines necessary to drive the LCD have been implemented in this project.

Further, this can be extended to have the following advanced features:

* Transition effects during pop-up and pop-out of message boxes.
* Video playback facility using two frame buffers.
* Touch Screen interface for a better user interaction with facilities viz., Pinch and Zoom, Free hand drawing.

**REFERENCE**

1. “ An introduction to non-os programming”, http://www.friendlyarm.net/forum/topic/611. .
2. “Development board arm 1810”, http://www.ti.com/product/arm810.
3. “ Stellaris ware graphics library standalone package”, http://www.ti.com/tools/sw.grl.