



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

NURSE CALL BUTTON USING DTMF

submitted by

Alisha Shah (17BCE0930)

Navya RG (17BCE2416)

Vandita Pandey (17BCE0711)

**Report submitted for the
Final Project Review of**

**Course Code: CSE2006 – Microprocessors and Interfacing
Slot: C1
Lab: L47 + L48**

Professor: Krishnamoorthy A

Acknowledgement

It is our privilege to express our sincere gratitude to our Microprocessors and interfacing faculty, Prof. Krishnamoorthy A for his valuable inputs, able guidance, encouragement, whole- hearted cooperation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our Head of Department and Dean for encouraging and allowing us to present the project on the topic -Nurse call button using DTMF.

We take this opportunity to thank all our lecturers who have directly or indirectly helped our project and the management for providing us with top quality infrastructure and lab assistant. We pay our love and respect to our parents and all other family members and friends for their love and encouragement throughout our education. Last but not the least we express our gratitude to our friends for their cooperation and support.

Table of Contents:

Abstract	4
Chapter-1	4
Aim	4
Objective	4
Introduction	4
Chapter-2	5
Literature Survey	5
Existing and Proposed System	5
Chapter-3	6
Proposed System Design Architecture	6
Architecture Explanation	6
DTMF:	6
CM8870PI DTMF Decoder:	8
Arduino Uno (ATMEGA328P):	10
Operation of the system:	14
Other requirements:	15
Algorithms & Pseudocode	15
Flowchart	15
Algorithm	15
Pseudocode	16
Chapter-4	18
Results & Discussion	18
Conclusion & Future Work	18
References	18

1. Abstract

A nurse call button makes use of Arduino interfaced with a DTMF decoder to send messages/email to the nurse when the patient presses a button on his phone. Each button on his phone corresponds to a specific message like need water, need food or emergency. When he presses any button Dual Tone Multi Frequency waves are sent. These waves are then decoded by the DTMF decoder which is connected to a microcontroller like arduino. When the microcontroller receives this data it takes it and sends messages using a nexmo account in addition to sending a mail. This methodology uses DTMF waves which is faster than sending messages over the internet. DTMF technology uses methods of display signal processing, thus increasing its range of working. DTMF modules generate sinusoidal tones, which will prevent misinterpretation of the harmonics, making it safer to use.

Keywords: DTMF, DTMF decoder, Dual Tone Multi Frequency, Mobile phone, Micro-controller, Nexmo

2. Chapter-1

2.1. Aim

To device an appliance which enables the user to communicate with their caretaker/ nurse with ease.

2.2. Objective

To implement a device using Arduino and DTMF decoder to aid the bedridden. Our project focuses on devising technology which would enable the patient to communicate with their caretakers/ nurse just by the press of a button.

2.3. Introduction

These days the overall health of Indian population is deteriorating due to excessive junk food, pollution, genetic diseases etc. To make patients feel better, they need to be empathized with and one way we can do that is using technology. As everyone owns a mobile phone these days, it can be achieved with ease. As a result of this people are often in and out of hospitals. This increases the workload of doctors and nurses. A simple nurse call button would be a very useful tool to remove some of the weight off of their shoulders. The patient would be able to specify which service he requires and the nurse will get a message/email containing the request along with the other requests from other patients. If a nurse receives a message saying the patient is in severe pain, she would have to rush to his room even though she might have gotten messages from other patients. Or if the nurse is on a particular floor, she can attend to all the patients on that floor before moving on to the next using these messages.

3. Chapter-2

3.1. Literature Survey

The earliest version of nurse call button was adapted from the bell systems for servants in affluent homes. They were first used by Florence Nightingale. Initially in its simplest form a nurse call button consists of a push button in the patient's room and a light outside the patient's door would glow. There was also a central location where all the calls could be viewed. With hospitals becoming larger, healthcare facilities needed a different system rather than the push button. With the evolution of technology, wireless nurse call systems, systems using DTMF have become easy to use and maintain.

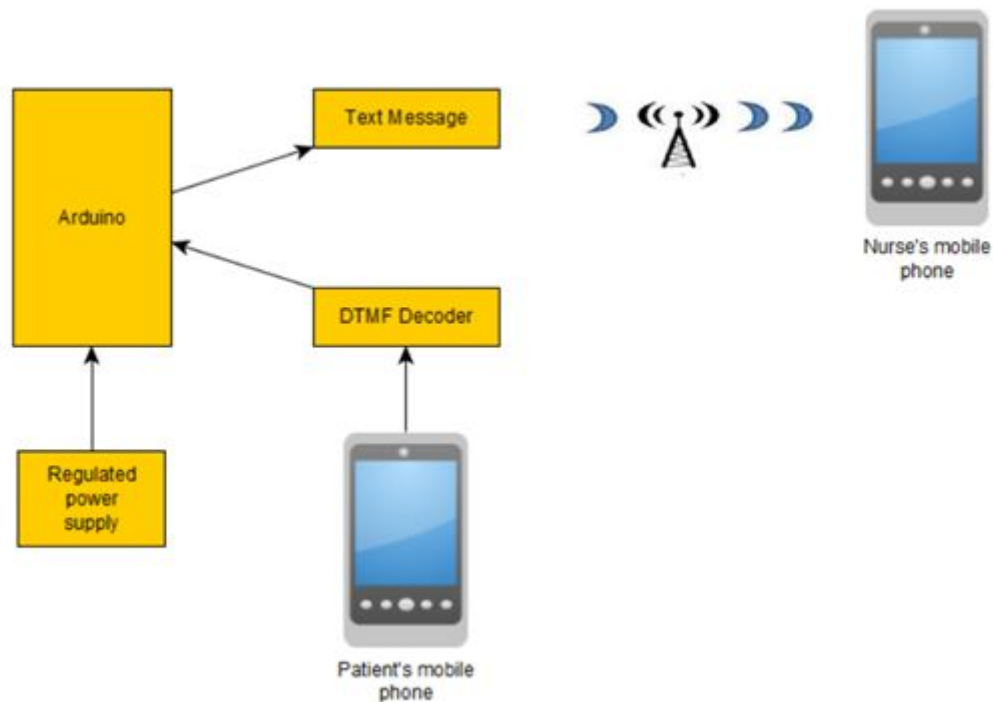
3.2. Existing and Proposed System

Several nurse call buttons exist which on pushing a button, switches on a light outside the patient's room indicating that the nurse's services are required by the patient. The nurse on seeing the light switched on can go to the patient's room to offer help. But here, the nurse wouldn't know what the patient is calling him/her for. So the nurse would have to rush to the patient's room thinking it might be something serious. In cases where more than one patient calls the nurse, she would have to prioritize between all the patients and offer help to the most serious issue first.

The above-mentioned system is archaic and can be upgraded using technology. We propose a system where the patient can select a particular service he requires from a set of buttons (example: need food, need water, call the doctor, room cleaning etc). When the patient pushes this button, a message is sent to the nurse's mobile phone indicating the specific service that the patient requires. This way when more than one patient calls then nurse, she would be able to give preference to the most serious issue first.

4. Chapter-3

4.1. Proposed System Design Architecture



The patient's cell phone is connected to a DTMF receiver. When the patient presses any key on his keypad, a DTMF tone is generated which is sent to the DTMF receiver through an audio jack. This tone is filtered and then decoded and sent to the Arduino board. Pins 4-7 on the DTMF decoder are connected to pins Q4-1 on the Arduino board. This 4-bit BCD number can have upto 16 values Each value is assigned a message and sent to the receiver over email or text.

4.2. Architecture Explanation

DTMF:

Dual Tone Multi Frequency is used in telecommunication over analog communication lines. They are used for signalling in the voice frequency band from telephone handsets and other communication devices and vice versa. The DTMF technology makes use of signals of eight different frequencies. They are transmitted in pairs to represent sixteen values which may be numbers, symbols or letters. When a user presses a key of the handset a DTMF signal is generated. This signal is a one of a kind tone consisting of two different frequencies, one from the higher frequency range ($>1\text{KHz}$) and another from the lower frequency range ($<1\text{KHz}$). The resultant tone is the convolution of the two frequencies and thus is composed of two pure sine waves of low and high frequencies superimposed on each other. The two frequencies explicitly represent the digits on the telephone keypad.

The generated signal can be mathematically expressed as

$$f(t) = A_H \sin(2\pi f_H t) + A_L \sin(2\pi f_L t)$$

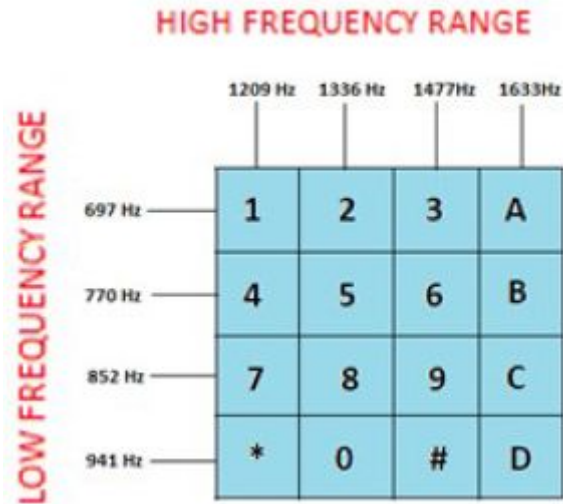
Where:-

A_H, A_L : Amplitudes

f_H : High frequency range

f_L : Low frequency range

The high and low range frequency for each key on the keypad has been shown in the figure below.



From the figure we can see that each key can uniquely be identified by its row and column. For Example:- If key “4” is pressed the dual tone frequency would be $770 + 1209 = 1979\text{Hz}$.

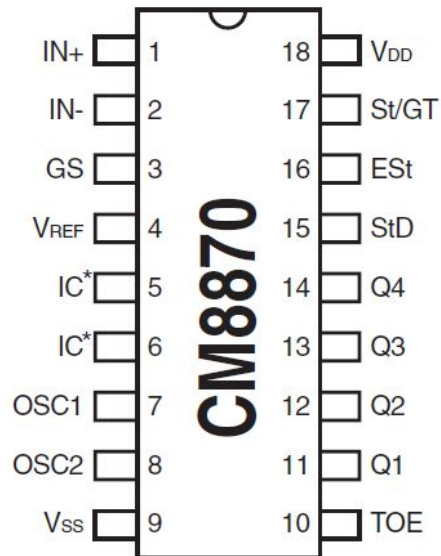
Keys used in the nurse call button:

Key	Low Freq	High Freq	Q4	Q3	Q2	Q1
1	679	1209	0	0	0	1
2	679	1336	0	0	1	0
3	679	1477	0	0	1	1
4	770	1209	0	1	0	0
5	770	1336	0	1	0	1
6	770	1477	0	1	1	0
7	852	1209	0	1	1	1
8	852	1336	1	0	0	0
9	852	1477	1	0	0	1

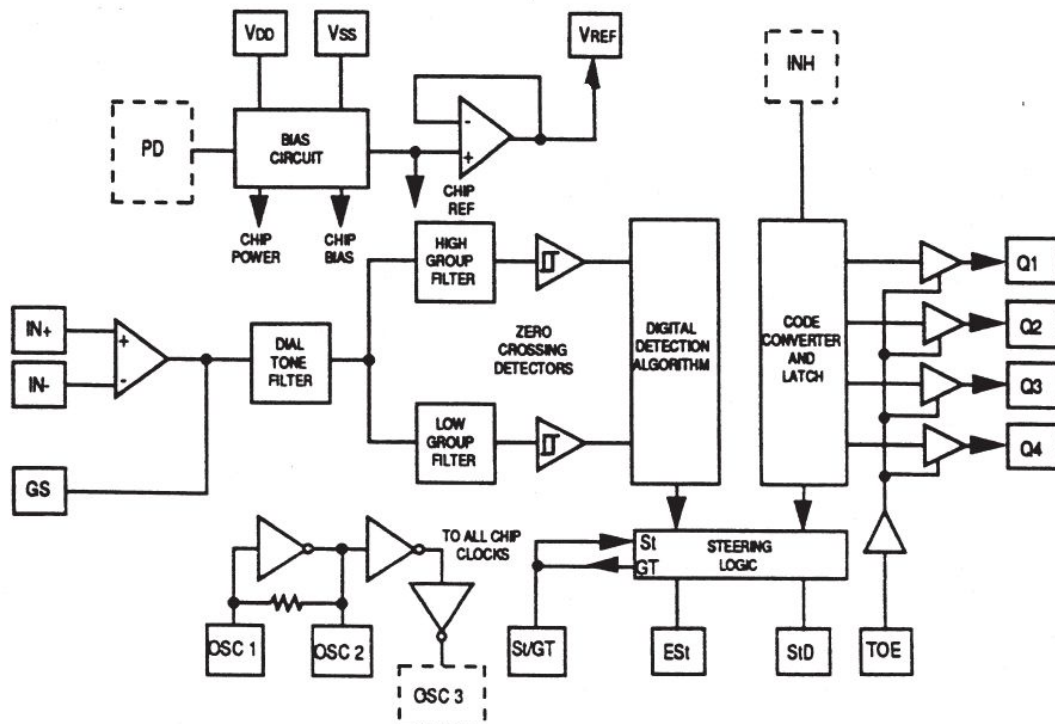
0	941	1209	1	0	1	0
*	941	1336	1	0	1	1

CM8870PI DTMF Decoder:

Pin Diagram:



Block Diagram:



The CAMD CM8870/70C provides full DTMF receiver capability by integrating both the bandsplit filter and digital decoder functions into a single 18-pin DIP, SOIC, or

20-pin PLCC package. The CM8870/70C is manufactured using state-of-the-art CMOS process technology for low power consumption (35mW, max.) and precise data handling. The filter section uses a switched capacitor technique for both high and low group filters and dial tone rejection. The CM8870/70C decoder uses digital counting techniques for the detection and decoding of all 16 DTMF tone pairs into a 4-bit code. This DTMF receiver minimizes external component count by providing an on-chip differential input amplifier, clock generator, and a latched three-state interface bus. The on-chip clock generator requires only a low cost TV crystal or ceramic resonator as an external component.

Band Split Filter: A single tone is superimposed with two frequencies, one is low and the other is high frequency. Band split filter is used to split these two frequencies into high group frequency and low group frequency, Then these frequencies have to be decoded. Separation of the low-group and high-group tones is achieved by applying the dual-tone signal to the inputs of two 9th-order switched capacitor bandpass filters. The bandwidths of these filters correspond to the bands enclosing the low-group and high-group tones. The filter section also incorporates notches at 350 Hz and 440 Hz which provides excellent dial tone rejection. Each filter output is followed by a single order switched capacitor section that smooths the signals prior to limiting. Signal limiting is performed by high gain comparators. These comparators are provided with hysteresis to prevent detection of unwanted low-level signals and noise. The outputs of the comparators provide full-rail logic swings at the frequencies of the incoming tones.

Decoding: Its decoder uses digital counting techniques to detect and decode all 16 DTMF tone pairs into a 4-bit code. It determines the frequencies of the limited tones and to verify that these tones correspond to standard DTMF frequencies. A complex averaging algorithm is used to protect against tone simulation by extraneous signals (such as voice) while providing tolerance to small frequency variations. The averaging algorithm has been developed to ensure an optimum combination of immunity to “talk-off” and tolerance to the presence of interfering signals (third tones) and noise. When the detector recognizes the simultaneous presence of two valid tones (known as “signal condition”), it raises the “Early Steering” flag (ESt). Any subsequent loss of signal condition will cause ESt to fall.

Pin description:

PIN FUNCTION		
Name	Description	
IN+	Non-inverting Input	Connection to the front-end differential amplifier
IN-	Inverting Input	
GS	Gain Select	Gives access to output of front-end differential amplifier for connection of feedback resistor.
V _{REF}	Reference voltage output (nominally V _{DD} /2). May be used to bias the inputs at mid-rail.	
INH	Inhibits detection of tones represents keys A, B, C, and D	
OSC3	Digital buffered oscillator output.	
PD	Power Down	Logic high powers down the device and inhibits the oscillator.
OSC1	Clock Input	3.579545 MHz crystal connected between these pins completes internal oscillator.
OSC2	Clock Output	
V _{SS}	Negative power supply (normally connected to OV).	
TOE	Three-state output enable (input). Logic high enables the outputs Q ₁ -Q ₄ . Internal pull-up.	
Q ₁ Q ₂ Q ₃ Q ₄	Three-state outputs. When enabled by TOE, provides the code corresponding to the last valid tone pair received. (See Fig. 2).	
StD	Delayed steering output. Presents a logic high when a received tone pair has been registered and the output latch is updated. Returns to logic low when the voltage on St/GT falls below V _{TSt} .	
ES _t	Early steering output. Presents a logic high immediately when the digital algorithm detects a recognizable tone pair (signal condition). Any momentary loss of signal condition will cause ES _t to return to a logic low.	
St/G _t	Steering input/guard time output (bidirectional). A voltage greater than V _{TSt} detected a St causes the device to register the detected tone pair. The GT output acts to reset the external steering time constant, and its state is a function of ES _t and the voltage on St. (See Fig. 2)	
V _{DD}	Positive power supply.	
IC	Internal Connection.	Must be tied to V _{SS} (for 8870 configuration only)

Arduino Uno (ATMEGA328P):

The Arduino Uno is a microcontroller for which ATmega328 (datasheet) forms the base. It has 6 analog inputs, 14 digital I/O pins (6 of the 14 pins can be used as PWM outputs), a USB connection, a 16 MHz ceramic resonator, a reset button, an ICSP header and a power jack.

ATMEGA328p is a single chip, 32K 8-bit microcontroller base on AVR architecture. ATMEGA328p is mostly used in many projects and autonomous systems where simple, low powered, low cost microcontroller is needed.

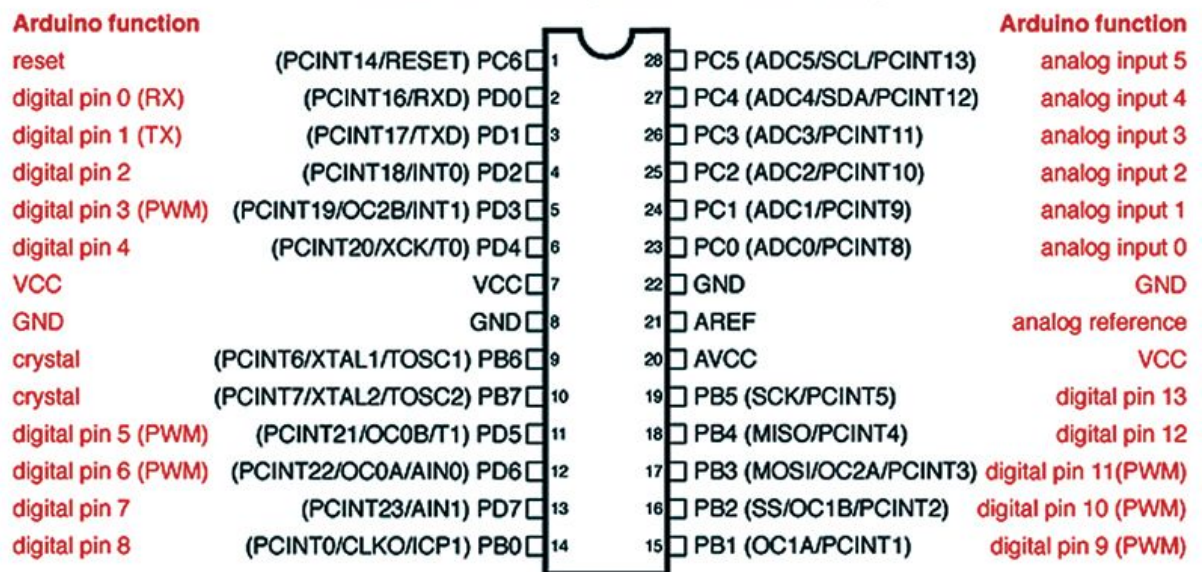
A summary of Arduino Uno is given below:

- Microcontroller : ATmega328 (high-performance Atmel AVR RISC-based microcontroller)
- Operating Voltage : 5V
- Input Voltage (recommended) : 7-12V
- Input Voltage (limits) : 6-20V
- Digital I/O Pins : 14 [PWM output is provided by 6 of them]
- Analog Input Pins : 6
- DC Current per I/O Pin : 40 mA

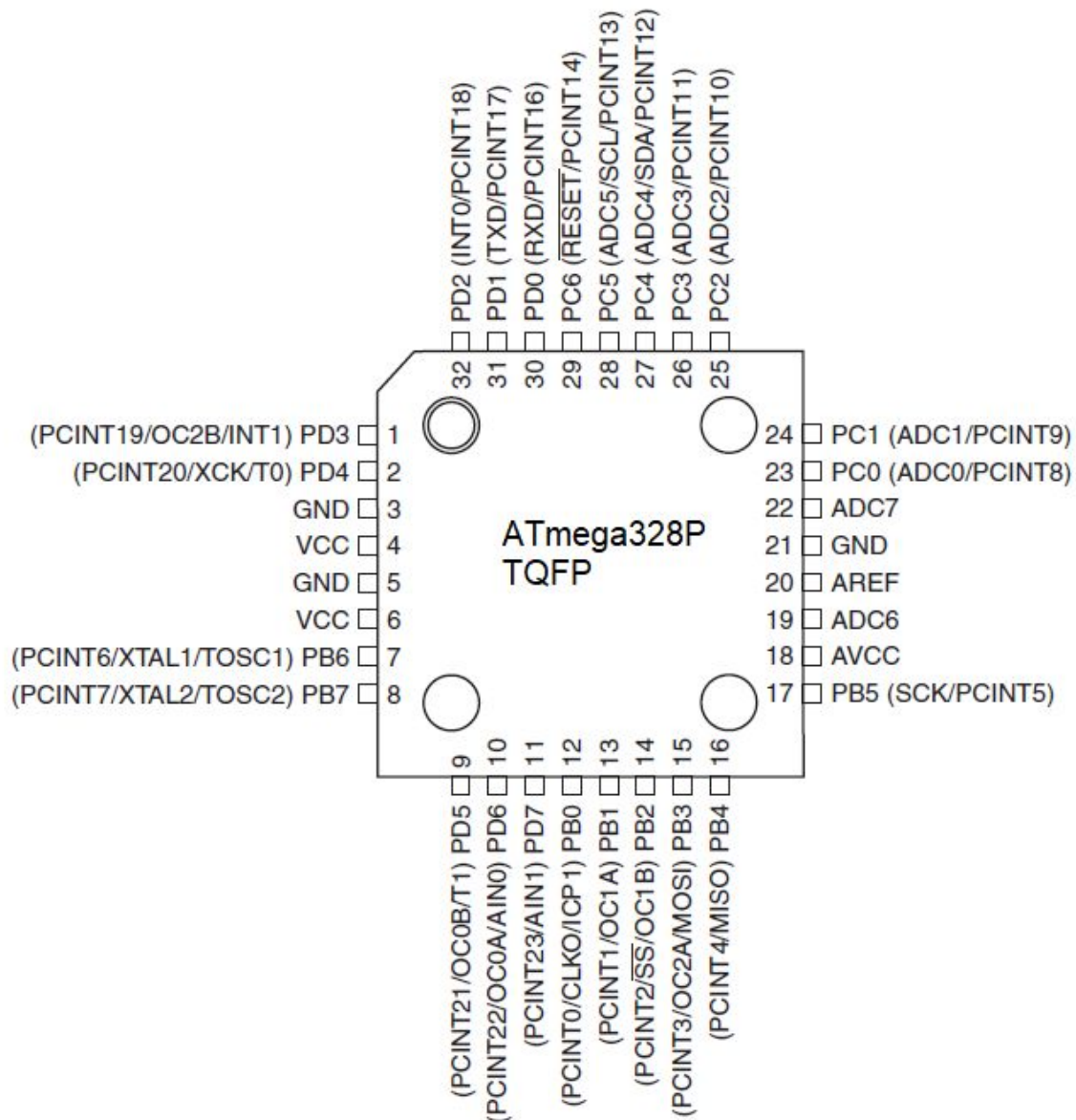
- DC Current for 3.3V Pin : 50 mA
- Flash Memory : 32 KB [ATmega328]; 0.5 KB of it is used by Bootloader (read-while write capability)
- SRAM : 2 KB [ATmega328]
- EEPROM : 1 KB [ATmega328]
- Clock Speed : 16 MHz
- General purpose I/O lines : 23
- General purpose working registers : 32
- Timer/counters with compare modes : 3
- Throughput : 1MIPS per MHz
- Internal and external interrupts
- Serial programmable USART
- SPI serial port
- Five software selectable power saving modes
- A byte-oriented 2-wire serial interface
- 10-bit 6-channel analog to digital converter (8-channels in QFN/MLF and TQFP packages)
- Programmable watchdog timer with internal oscillator

Pin Diagram:

Arduino ATmega328 Pin Mapping



Digital Pins 11, 12 & 13 are used by the ICSP header for MISO, MOSI, SCK connections (Atmega168 pins 17, 18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.



Pin Description of ATmega328P:

V_{CC}	Digital supply voltage
GND	Ground
Port B ($PB_{7:0}$) XTAL ₁ /XTAL ₂ /TOSC ₁ /TOSC ₂	<p>Port B is an 8-bit bi-directional I/O port. It also has an internal pull-up resistor selected for each bit. It's output buffers have symmetrical drive characteristics made up of both high sink and source capability. When Port B is configured as an input port, its pins that are externally pulled low will source current if the pull-up resistors are activated. The pins of Port B are tri-stated when a reset condition becomes active, even if the clock is not running.</p> <p>PB6 can be used as input to the internal clock</p>

	<p>operating circuit and input to the inverting Oscillator-amplifier depending on the clock selection fuse settings.</p> <p>PB7 can be set as the output from the inverting Oscillator-amplifier depending on the clock selection fuse settings.</p> <p>If the AS₂ bit in ASSR is set, PB₇₋₆ is used as TOSC₂₋₁ input for the Asynchronous Timer/Counter2, if the Internal Calibrated RC Oscillator is used as chip clock source.</p>
Port C (PC _{5:0})	<p>Port C is a 7-bit bi-directional I/O port. It also has an internal pull-up resistor for each bit. The PC₅₋₀ output buffers have symmetrical drive characteristics with both source capability and high sink. If the pull-up resistors are activated, Port C pins that are externally pulled low will source current when Port C is set as input. Even if the clock is not running, when a reset condition becomes active, the Port C pins are tri-stated.</p>
PC ₆ / \overline{RESET}	<p>PC6 is used as an input pin if the RSTDISBL fuse is programmed, otherwise it is used as a Reset input. Even when the clock is not running, a Reset is generated when there is a low level on this pin for longer than the minimum pulse length. Shorter pulses generating a Reset is not guaranteed.</p>
Port D (PD _{7:0})	<p>Port D is an 8-bit bi-directional I/O port. It also has an internal pull-up resistor for each bit.. The Port D output buffers have symmetrical drive characteristics with both source capability and high sink. If the pull-up resistors are activated, Port D pins that are externally pulled low will source current when Port D is set as input. Even if the clock is not running, when a reset condition becomes active, the Port D pins are tri-stated..</p>
AV _{CC}	<p>AV_{CC} acts as the supply voltage pin for the analog to digital converter, PC_{3:0} and ADC_{7:6}. Even if the analog to digital converter is not used, it should be externally connected to V_{CC}. It should be connected to V_{CC} through a low-pass filter if the ADC is used. Note that PC_{6..4} make use of digital supply voltage, V_{CC}.</p>

AREF	Analog reference pin for ADC.
ADC _{7:6} (TQFP and QFN/MLF Package Only)	ADC _{7:6} serve as analog inputs to the A/D converter in the TQFP and QFN/MLF package. Power for these pins is supplied by the analog supply and can be used as 10-bit ADC channels.

Operation of the system:

The DTMF tone is sent from the patient's (sender) mobile phone to the DTMF module on pressing any key. This connection is achieved by connecting a 3.5mm audio jack to the sender's phone from the DTMF module. The tone sent is received by the DTMF receiver and then it is filtered into high and low group frequency. This tone is then sent to the DTMF decoder to be converted into a 4-bit BCD value. This 4-bit value is sent to the Arduino. The 4-bit BCD value is output through Q4, Q3, Q2, Q1 of MT8870 Decoder IC which are connected to pins 4,5,6,7 of ATmega328P Microcontroller respectively. A specific message is assigned to each combination of values in pins 4-7 of Arduino. This is done by uploading a program onto the Arduino Uno using the Arduino software.

The Arduino board is connected to one of the USB ports of a laptop. A python program is used to take the input received from the same port. This input is decoded and the corresponding message is sent as an email as well as a text message using Nexmo and Gmail respectively.

Messages corresponding to each key is given below:

Key	Message
1	Emergency in patient's room
2	Patient is hungry
3	Patient is thirsty
4	Patient requires medicines
5	Patient is in pain
6	Patient needs to use the restroom
7	Patient needs change of clothes
8	Patient need to see the doctor

9	Patient's room requires cleaning
0	Patient needs to meet family members

Other requirements:

Arduino Desktop IDE was used to program the Arduino Board.

Python2.7 was used to send messages and emails.

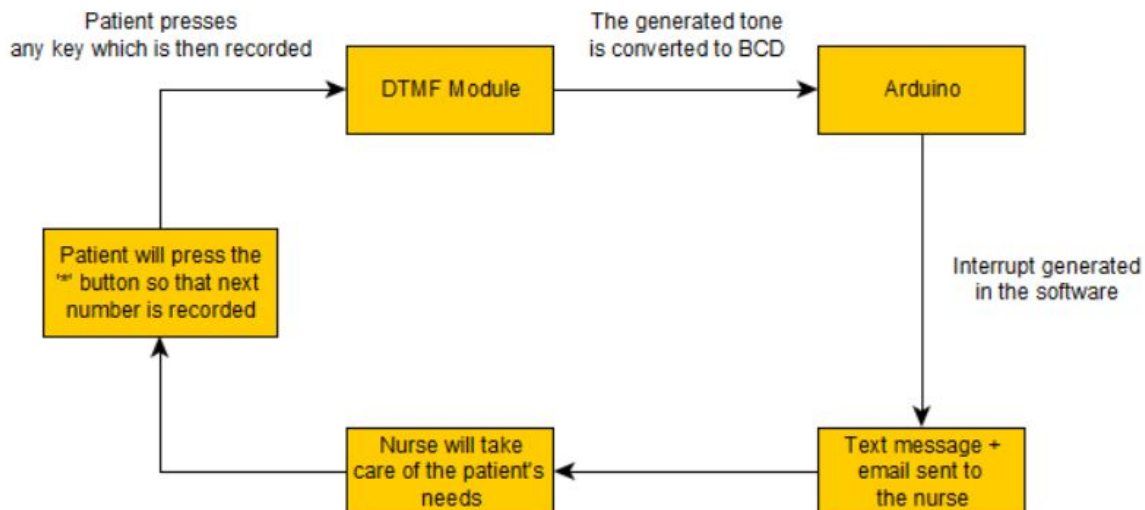
Two mobile phones are required (Patient's and nurse's).

Connecting wires are required to interface DTMF decoder with Arduino.

A 3.5 mm audio jack is used to connect the sender's cell phone to a DTMF decoder.

4.3. Algorithms & Pseudocode

Flowchart



Algorithm

Algorithm of arduino program for reading and processing input:

- Assign the value of characters a and b as '*' and '*'

Setup:

- Set speed of communication to 9600
- Set pin 8-11 as input pins

Loop:

- Read input from pins 8-11
- If it is equivalent to '*', then record the next input value from pin 8-11
- According to the values of pins 8-11 assign values to b as follows:

PIN 8	PIN 9	PIN 10	PIN 11	b
-------	-------	--------	--------	---

0	0	0	1	'1'
0	0	1	0	'2'
0	0	1	1	'3'
0	1	0	0	'4'
0	1	0	1	'5'
0	1	1	0	'6'
0	1	1	1	'7'
1	0	0	0	'8'
1	0	0	1	'9'
1	0	1	0	'0'

- For all the above values of pins 8-11, print the value of b in the serial monitor
- For all other values of pins 8-11, do not print the value of b in the serial monitor

Algorithm of python program for sending email/text message:

- Libraries to import:
Serial - for getting input from Arduino
Smtplib - for sending email
Nexmo - for sending text messages
- Get input from the communication port to which Arduino is connected
- Create an object with:
Keys - numbers on the patients keypad
Values - message each number corresponds to
- According to the input received from Arduino, get the corresponding message from the object that was created as the input will be one of the keys
- Send this message via email and text message.

Pseudocode

Pseudocode of arduino program for reading and processing input:

Char a = '*'

Char b = '*'

```

setup(){
    serial.begin(9600);
    pinMode(8, INPUT);
    pinMode(9, INPUT);
    pinMode(10, INPUT);
    pinMode(11, INPUT);

```



```

}
loop(){
    Input[] = {digitalRead(8), digitalRead(9), digitalRead(10), digitalRead(11)};
    if((input[0]==0 and input[1]==0 and input[2]==0 and input[3]==1)
        b='1';
    Else if((input[0]==0 and input[1]==0 and input[2]==1 and input[3]==0)
        b='2';
    Else if((input[0]==0 and input[1]==0 and input[2]==1 and input[3]==1)
        b='3';
    Else if((input[0]==0 and input[1]==1 and input[2]==0 and input[3]==0)
        b='4';
    Else if((input[0]==0 and input[1]==1 and input[2]==0 and input[3]==1)
        b='5';
    Else if((input[0]==0 and input[1]==1 and input[2]==1 and input[3]==0)
        b='6';
    Else if((input[0]==0 and input[1]==1 and input[2]==1 and input[3]==1)
        b='7';
    Else if((input[0]==1 and input[1]==0 and input[2]==0 and input[3]==0)
        b='8';
    Else if((input[0]==1 and input[1]==0 and input[2]==0 and input[3]==1)
        b='9';
    Else If (input[0]==1 and input[1]==0 and input[2]==1 and input[3]==0)
        b='0';
    Else if((input[0]==1 and input[1]==0 and input[2]==1 and input[3]==1)
        a='*'; b='*';

    if(a!=b and a=='*'){
        Serial.println(b);
        a=b;
    }
}

```

Pseudocode of python program for sending email/text message:

```

port='COM5'
baud=9400
input=serial.Serial(port, baud)
sender='sender@gmail.com'
password=sender's password'
receiver='receiver@gmail.com'
Smtplib_port=587
client = nexmo.Client(key='7fe56a9e', secret='qi1ltOKPRFKnU8rf')
msgs={
    1: Emergency in patient's room,
    2: Patient is hungry,
    3: Patient is thirsty,
    4: Patient requires medicines,

```

- 5: Patient is in pain,
- 6: Patient needs to use the restroom,
- 7: Patient needs change of clothes,
- 8: Patient needs to see the doctor,
- 9: Patient's room requires cleaning,
- 0: Patient needs to meet family members

```

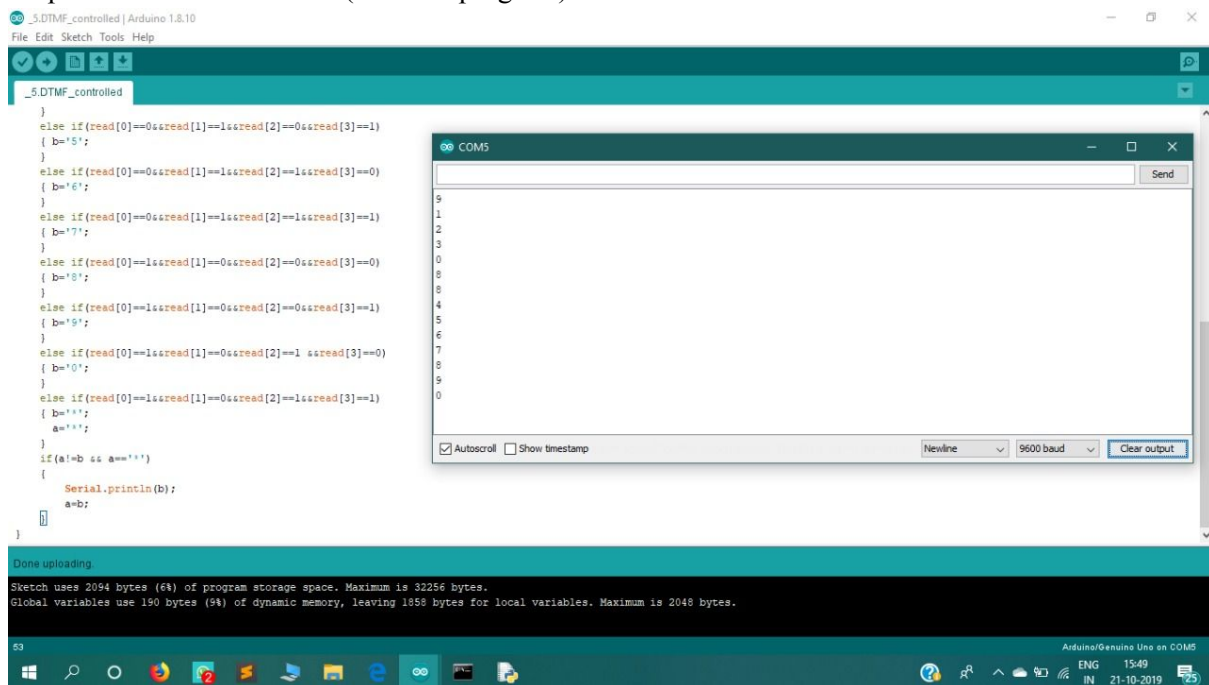
    }
While (true):
    in =input.readline().strip().decode('utf-8')
    print(in)
    msg=msgs.get(in, "Invalid input")
    client.send_message({
        'from': 'Nexmo',
        'to': '919080320171',
        'text': msg,
    });
    email=smtplib.SMTP('smtp.gmail.com', Smpt_port)
    email.starttls().login(sender, password).sendmail(sender, receiver, msg)

```

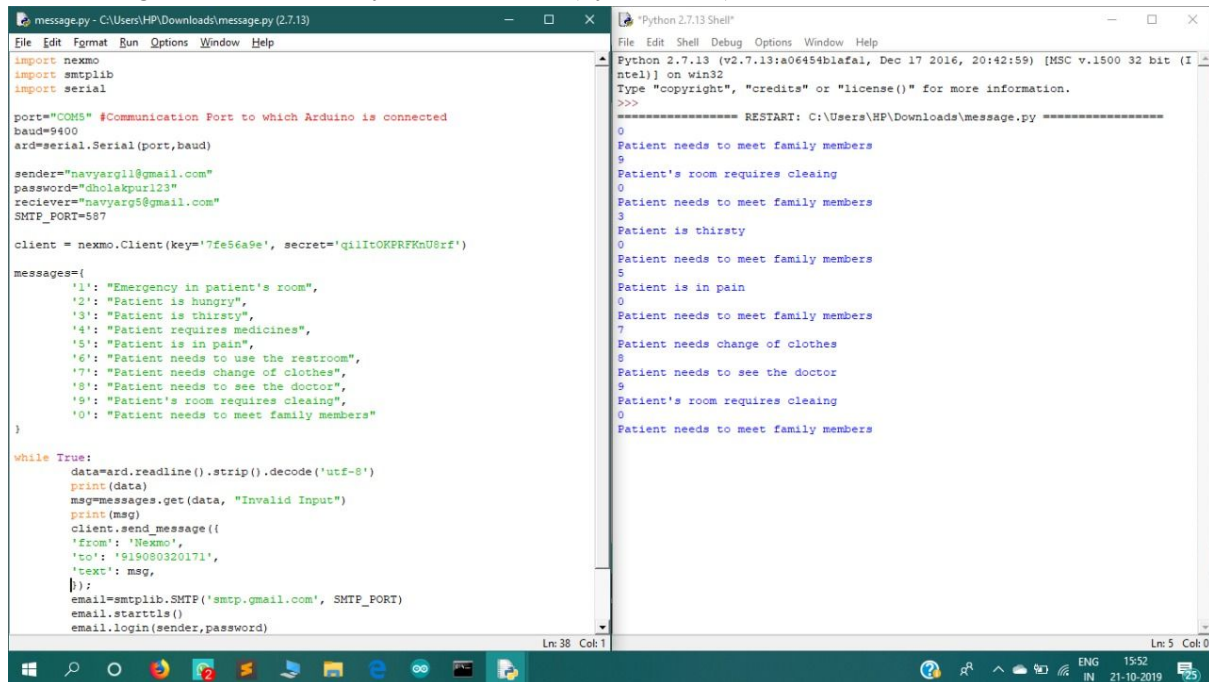
5. Chapter-4

5.1. Results & Discussion

Output on the serial monitor (Arduino program):



Text messages and emails set by sender-Patient (Python code):



The image shows a Windows desktop with two windows. The left window is a text editor titled 'message.py - C:\Users\HP\Downloads\message.py (2.7.13)'. It contains a Python script that uses the 'nexmo' library to send SMS messages. The script defines a list of messages and a loop that reads input from a serial port and sends the corresponding message via SMS. The right window is a 'Python 2.7.13 Shell' command prompt. It shows the execution of the script, which outputs a list of messages and then sends them via SMS. The messages are: 'Patient needs to meet family members', 'Patient's room requires cleaning', 'Patient is thirsty', 'Patient is in pain', 'Patient needs to meet family members', 'Patient needs change of clothes', 'Patient needs to see the doctor', 'Patient's room requires cleaning', and 'Patient needs to meet family members'.

```
message.py - C:\Users\HP\Downloads\message.py (2.7.13)
File Edit Format Run Options Window Help

import nexmo
import smtplib
import serial

port="COM5" #Communication Port to which Arduino is connected
baud=9600
ard=serial.Serial(port,baud)

sender="navyarg11@gmail.com"
password="dholakpur123"
receiver="navyarg5@gmail.com"
SMTP_PORT=587

client = nexmo.Client(key='7fe56a9e', secret='q11tOKPRFKnUSrf')

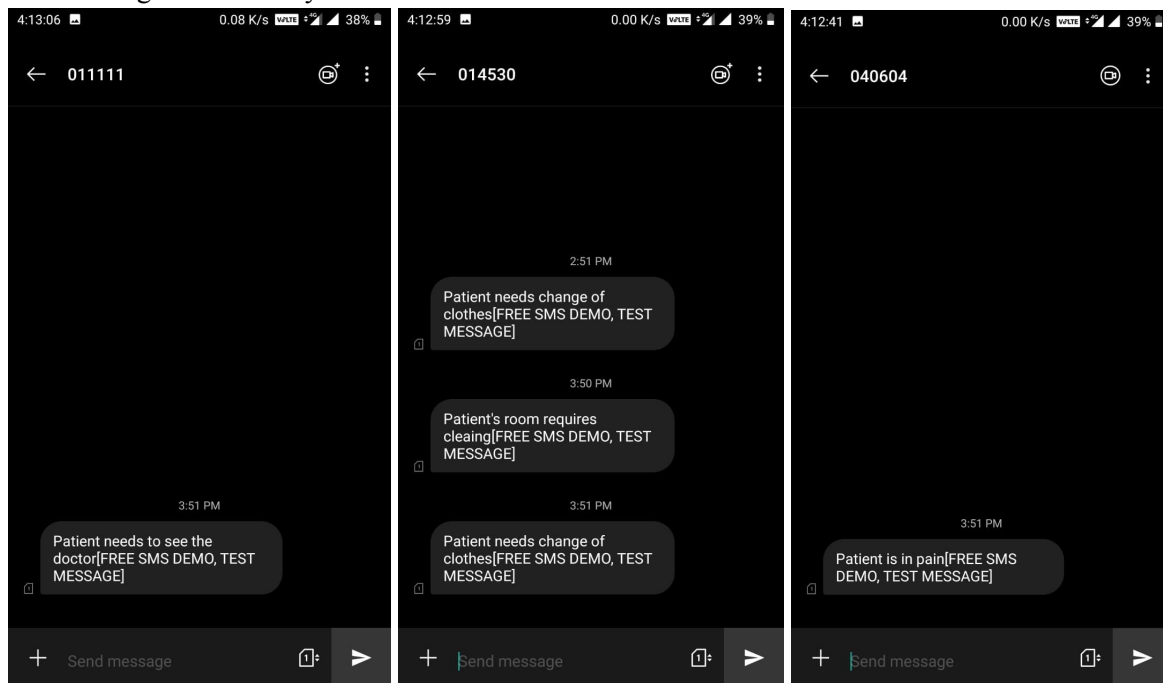
messages=[
    '1: "Emergency in patient's room",
    '2: "Patient is hungry",
    '3: "Patient is thirsty",
    '4: "Patient requires medicines",
    '5: "Patient is in pain",
    '6: "Patient needs to use the restroom",
    '7: "Patient needs change of clothes",
    '8: "Patient needs to see the doctor",
    '9: "Patient's room requires cleaning",
    '0: "Patient needs to meet family members"
]

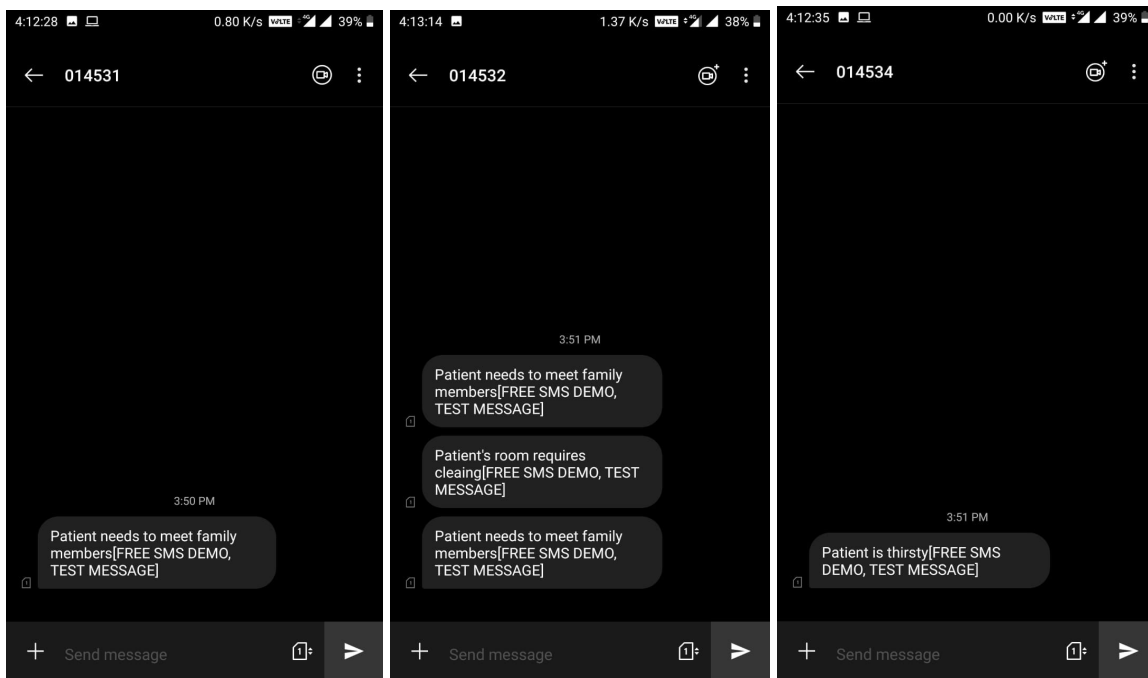
while True:
    data=ard.readline().strip().decode('utf-8')
    print(data)
    msg=messages.get(data, "Invalid Input")
    print(msg)
    client.send_message({
        'from': 'Nexmo',
        'to': '919080320171',
        'text': msg,
    })
    email=smtplib.SMTP('smtp.gmail.com', SMTP_PORT)
    email.starttls()
    email.login(sender,password)

Python 2.7.13 Shell
File Edit Shell Debug Options Window Help

Python 2.7.13 (v2.7.13:a06454b1afai, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\HP\Downloads\message.py =====
0
Patient needs to meet family members
9
Patient's room requires cleaning
0
Patient needs to meet family members
3
Patient is thirsty
0
Patient needs to meet family members
5
Patient is in pain
0
Patient needs to meet family members
7
Patient needs change of clothes
8
Patient needs to see the doctor
9
Patient's room requires cleaning
0
Patient needs to meet family members
```

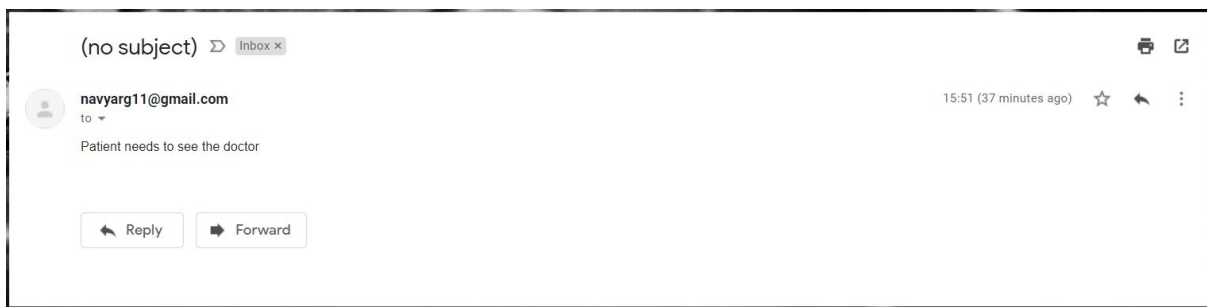
Text messages received by receiver-Nurse:





Emails received by receiver-Nurse:





Arduino program:

```
int Q4=8;
int Q3=9;
int Q2=10;
int Q1=11;
char a='*';
char b='*';

void setup()
{
  Serial.begin(9600);
  pinMode(Q4,INPUT);
  pinMode(Q3,INPUT);
  pinMode(Q2,INPUT);
  pinMode(Q1,INPUT);
}

void loop()
{
  int read[]={digitalRead(Q4),digitalRead(Q3),digitalRead(Q2),digitalRead(Q1)};
  if(read[0]==0&&read[1]==0&&read[2]==0&&read[3]==1)
  {
    b='1';
  }
}
```

```

else if(read[0]==0&&read[1]==0&&read[2]==1&&read[3]==0)
{ b='2';
}
else if(read[0]==0&&read[1]==0&&read[2]==1&&read[3]==1)
{ b='3';
}
else if(read[0]==0&&read[1]==1&&read[2]==0&&read[3]==0)
{ b='4';
}
else if(read[0]==0&&read[1]==1&&read[2]==0&&read[3]==1)
{ b='5';
}
else if(read[0]==0&&read[1]==1&&read[2]==1&&read[3]==0)
{ b='6';
}
else if(read[0]==0&&read[1]==1&&read[2]==1&&read[3]==1)
{ b='7';
}
else if(read[0]==1&&read[1]==0&&read[2]==0&&read[3]==0)
{ b='8';
}
else if(read[0]==1&&read[1]==0&&read[2]==0&&read[3]==1)
{ b='9';
}
else if(read[0]==1&&read[1]==0&&read[2]==1 &&read[3]==0)
{ b='0';
}
else if(read[0]==1&&read[1]==0&&read[2]==1&&read[3]==1)
{ b='*';
  a='*';
}
if(a!=b && a=='*')
{
  Serial.println(b);
  a=b;
}
}

```

Python program:

```

import nexmo
import smtplib
import serial

```

```

port="COM5" #Communication Port to which Arduino is connected
baud=9400
ard=serial.Serial(port,baud)

```

```
sender="navyarg11@gmail.com"
password="XXXX"
receiver="navyarg5@gmail.com"
SMTP_PORT=587
```

```
client = nexmo.Client(key='7fe56a9e', secret='qi1ItOKPRFKnU8rf')
```

```
messages={
    '1': "Emergency in patient's room",
    '2': "Patient is hungry",
    '3': "Patient is thirsty",
    '4': "Patient requires medicines",
    '5': "Patient is in pain",
    '6': "Patient needs to use the restroom",
    '7': "Patient needs change of clothes",
    '8': "Patient needs to see the doctor",
    '9': "Patient's room requires cleaning",
    '0': "Patient needs to meet family members"
}
```

```
while True:
    data=ard.readline().strip().decode('utf-8')
    print(data)
    msg=messages.get(data, "Invalid Input")
    print(msg)
    client.send_message({
        'from': 'Nexmo',
        'to': '919080320171',
        'text': msg,
    });
    email=smtplib.SMTP('smtp.gmail.com', SMTP_PORT)
    email.starttls()
    email.login(sender,password)
    email.sendmail(sender, receiver, msg)
```

Pins Q1-Q4 in the DTMF decoder were connected to pins 4-7 in the Arduino Uno board. The dual tone output from the patient's phone is sent to the DTMF module via an audi jack. This output was decoded into a 4-bit BCD value. The programmed arduino board receives this output through pins 4-7 which is then used for processing and sending messages/emails according to the key which was pressed.

This nurse call button proved to be of great help to nurses and caretakers who have multiple patients to take care of. This device has made it easier to take care of elderly people. Family members of the patients are also benefited from this device.

5.2. Conclusion & Future Work

The nurse call button proves to be very helpful in hospitals where a nurse would have to rush to each room when she is called. This simple tool would help nurses get to patient's rooms faster in case of an emergency to avoid any fatal incidents.

Every innovation can be upgraded. If not now, then some time in the near future. In the future, this nurse call button can be voice activated. When the patient needs something, he can press a button and say a particular phrase or keyword. This phrase can be assigned a particular frequency which would be generated and received by the DTMF receiver. This feature can be integrated with google assistant too. The patient would just have to say "Ok Google, I need water". Google assistant can be integrated with an app which would convert the message into a tone of a particular frequency. There are existing apps like IFTTT which could possibly integrate this feature.

6. References

Schenker, L (1960), "Pushbutton Calling with a Two-Group Voice-Frequency Code" (PDF), *The Bell System Technical Journal*, 39 (1): 235–255, ISSN 0005-8580

Gundanna, Veda, and Leo J. Agrillo. "DEFINITY* wireless business system—Health care application: nurse call system integration." *Bell Labs technical journal* 5.4 (2000): 171-184.

Woicke, Michael Duane. "Patient communication and monitoring method and system." U.S. Patent No. 9,191,485. 17 Nov. 2015.