----------MPI MINI PROJECT----------

Implementation of **Piano (8 keys)** (when a key is pressed, the corresponding frequency will be played) by using **8051 Microcontroller.**

A Project Report

Submitted in fulfilment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

In

Computer Science & Engineering

By

K.S.S.Keerti

1602-17-733-043

A.V.L.Likhita

1602-17-733-058

Under the guidance of

Mrs. B.Syamala

ASSISTANT PROFESSOR

Department of Computer Science & Engineering

Vasavi College of Engineering.

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad.

CONTENTS PAGE NO.

ACKNOWLEDGEMENT

I would like to express our heartfelt gratitude to Mrs. B.Syamala, our project guide, for his valuable guidance and constant support, along with his capable instruction and persistent encouragement.

I am grateful to our Head of Department, Dr. T. Adilakshmi, for her steady support and the provision of every resource required for the completion of this project.

I would like to take this opportunity to thank our Principal, Dr. S. V. RAMANA, as well as the management of the institute, for having designed an excellent learning atmosphere.

LIST OF FIGURES:

INTRODUCTION TO 8051 MICROCONTROLLER

8051 is an 8-bit family of microcontroller developed by Intel in the year 1981. This is one of the most popular family of microcontroller being used all across the world. This microcontroller was also referred as "system on a chip" because it has 128 bytes of RAM, 4Kbytes of ROM, 2 Timers, 1 Serial port, and four ports on a single chip. The CPU can work for only 8bits of data at a time because 8051 is an 8-bit processor. In case the data is larger than 8 bits then it has to be broken into parts so that the CPU can process conveniently. Most manufacturers have put 4Kbytes of ROM even though the quantity of ROM can be exceeded up to 64 K bytes. The 8051 has been in use in a wide number of devices, mainly because it is easy to integrate into a project or build a device around.

Since the basic layout of a microcontroller includes a CPU, ROM, RAM, etc. the 8051 microcontroller also has a similar layout. The following image shows a brief layout of a typical 8051 Microcontroller. It is a CISC based Microcontroller with Harvard Architecture (separate program and data memory).

8051 Internal Architecture

- 8 – bit CPU with two Registers A (Accumulator) and B.
- Internal ROM of 8K Bytes – It is a flash memory that supports in – system programming.
- Internal RAM of 256 Bytes – The first 128 Bytes of the RAM i.e. 00H to 7FH is again divided in to 4 banks with 8 registers (R0 – R7) in each bank, 16 bit addressable registers and 80 general purpose registers. The higher 128 Bytes of the RAM i.e. 80H to FFH consists of SFRs or Special Function Registers. Using SFRs we can control different peripherals like Timers, Serial Port, all I/O Ports, etc.

- 32 I/O Pins (Input / Output Pins) – Arranged as 4 Ports: P0, P1, P2 and P3.
- 8- bit Stack Pointer (SP) and Processor Status Word (PSW).
- 16 – bit Program Counter (PC) and Data Pointer (DPTR).
- Two 16 – bit Timers / Counters – T0 and T1.
- Control Registers – SCON, PCON, TCON, TMOD, IP and IE.
- Serial Data Transmitter and Receiver for Full – Duplex Operation – SBUF.
- Interrupts: Two External and Three Internal.
- Oscillator and Clock Circuit.

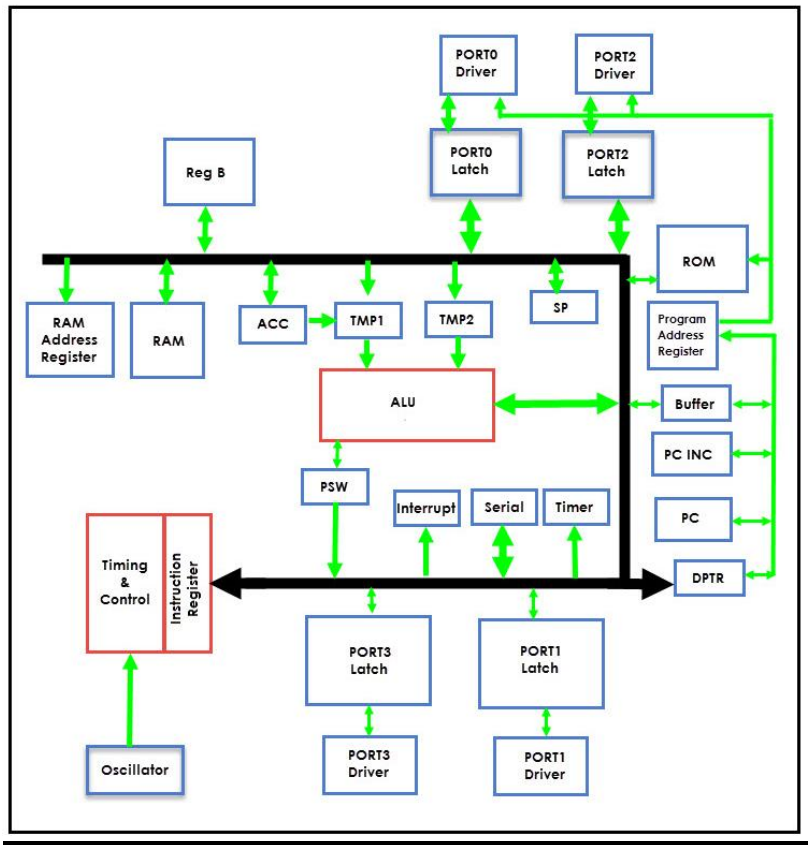


Figure 1

Pin Description of 8051 Microcontroller

- **Pins 1 to 8** − These pins are known as Port 1. This port doesn't serve any other functions. It is internally pulled up, bi-directional I/O port.
- **Pin 9** − It is a RESET pin, which is used to reset the microcontroller to its initial values.
- **Pins 10 to 17** − These pins are known as Port 3. This port serves some functions like interrupts, timer input, control signals, serial communication signals RxD and TxD, etc.
- **Pins 18 & 19** − These pins are used for interfacing an external crystal to get the system clock.
- **Pin 20** − This pin provides the power supply to the circuit.
- **Pins 21 to 28** − These pins are known as Port 2. It serves as I/O port. Higher order address bus signals are also multiplexed using this port.

- **Pin 29** − This is PSEN pin which stands for Program Store Enable. It is used to read a signal from the external program memory.
- **Pin 30** − This is EA pin which stands for External Access input. It is used to enable/disable the external memory interfacing.
- **Pin 31** − This is ALE pin which stands for Address Latch Enable. It is used to demultiplex the address-data signal of port.
- **Pins 32 to 39** − These pins are known as Port 0. It serves as I/O port. Lower order address and data bus signals are multiplexed using this port.
- **Pin 40** − This pin is used to provide power supply to the circuit.

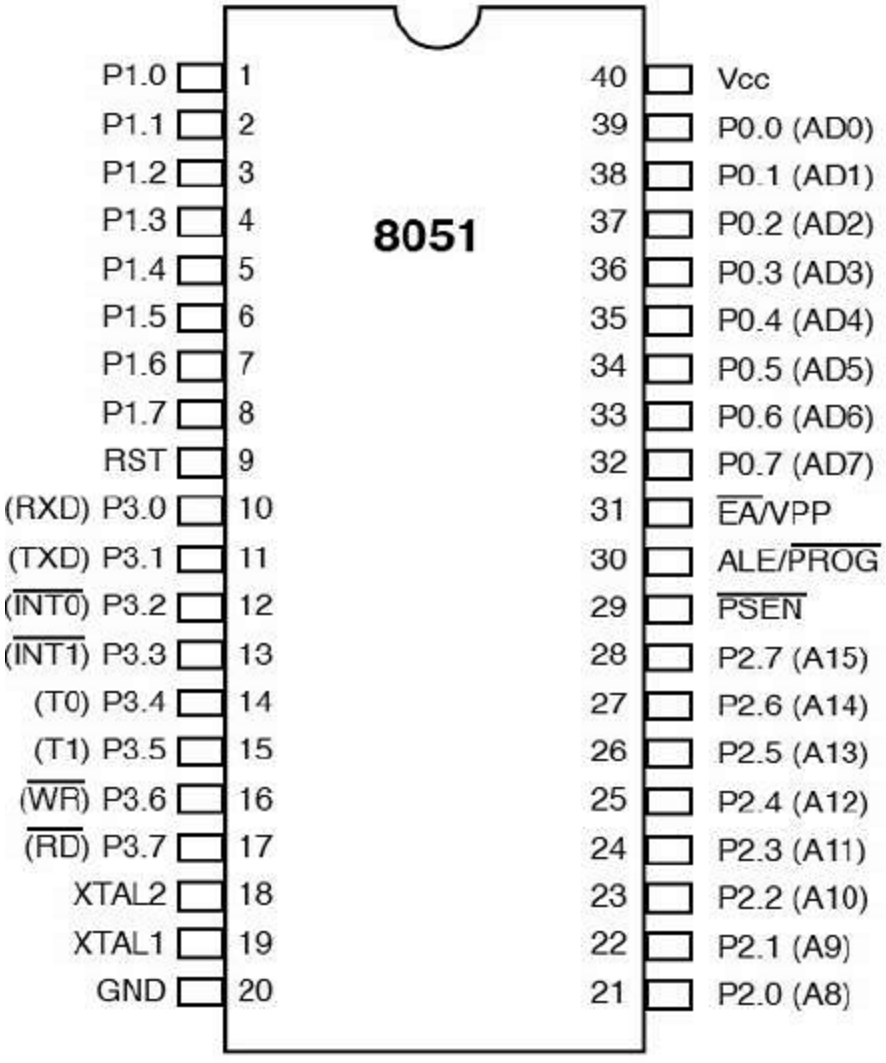


Figure 2

DESCRIPTION

The **piano** is a musical instrument played mainly by means of a keyboard. It is one of the most popular instruments in the world.

The rhythmic organization is based on rhythmic patterns called Taal. The melodic foundations are called ragas. One possible classification of ragas is into “melodic modes” or “parent scales”, known as Thaats, under which most ragas can be classified based on the notes they use.

Thaats may consist of up to seven scale degrees, or swara. Hindustani musicians name these pitches using a system called Sargam, the equivalent of Western movable do solfege:
Sa (Shadaj) = Do
Re (Rishab) = Re
Ga (Gandhar) = Mi
Ma (Madhyam) = Fa
Pa (Pancham) = So
Dha (Dhaivat) = La
Ni (Nishad) = Ti
Sa (Shadaj) = Do

Both systems repeat at the octave. The difference between sargam and solfege is that re, ga, ma, dha, and ni can refer to either “Natural” (Shuddha) or altered “Flat” (Komal) or “Sharp” (Tivra) versions of their respective scale degrees. As with movable do solfege, the notes are heard relative to an arbitrary tonic that varies from performance to performance, rather than to fixed frequencies, as on a xylophone.

In this project we are generating the same friquencies whch are excately same as **sa re ga ma pa dha ni sa** by using the controller.

SOFTWARE REQUIREMENT

For this project, we have used Keil MicroVision 3 as the software development environment. Keil MDK is the complete software development environment for a wide range of Arm Cortex-M based microcontroller devices. MDK includes the µVision IDE and debugger, Arm C/C++ compiler, and essential middleware components.

And for simulation, we have used Proteus 8 Professional (Schematic Capture). It allows us to develop hardware implementation of the project by providing a wide range of components to build circuits in workspace and thereby allows us to attach "hex" code to the simulation and run the simulation at real-time.

Features of Device:

Windows 10

64-bit Operating System

8 GB RAM

SIMULATION USING PROTEUS 8 :

The microcontroller used for this simulation is AT89C51 (ATMEL) microcontroller.

COMPONENTS:

1. AT89C51 Microcontroller - 8051 microcontroller – 4 KB Code, 33 MHz, 2 X 16-bit timers, UART

   This is the main component of the project, to the 8 pins belonging to Port 1 i.e. P1.0, P1.1, P1.2, P1.3, P1.4, P1.5, P1.6 and P1.7, we connect the 8 buttons. Upon the press of any button accordingly frequencies are generated, depending on the program written in the Keil software development tool.
2. Button – SPST Push button

   8 buttons are used to simulate 8 different keys of the Piano.
3. Speaker – Generic Load Speaker

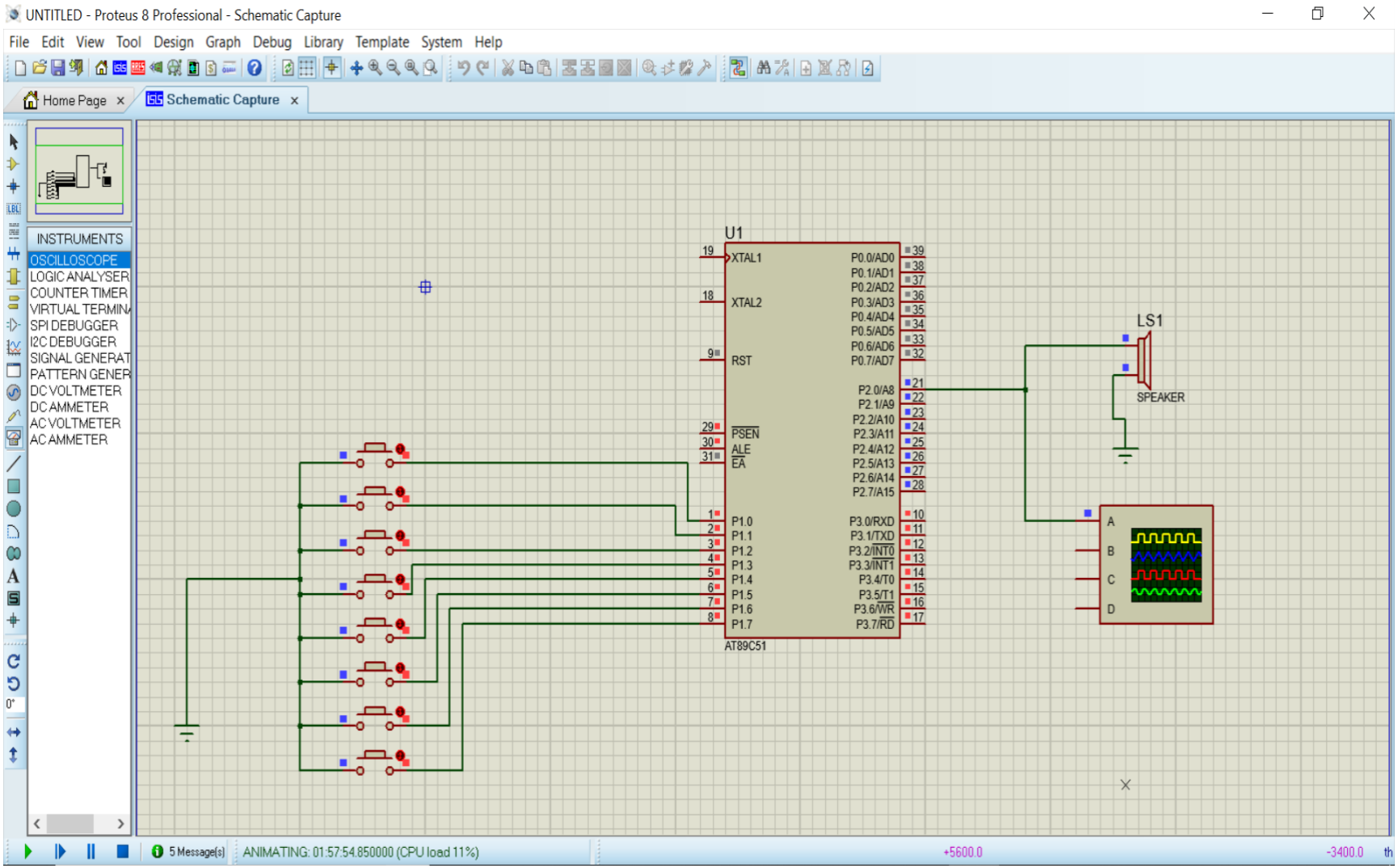
   The speaker acts as the audio output.
4. Oscilloscope

   The oscilloscope is used to display and analyse the waveform of electric signals. To identify and differentiate different frequencies produced by the speaker on different press of buttons.
5. Terminals – Ground

   The other ends of buttons are connected to the ground terminal.
6. Wires

   Wires have been used to connect all the components.

Figure 3


UNTITLED - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Home Page
Schematic Capture
INSTRUMENTS
OSCILLOSCOPE
LOGIC ANALYSER
COUNTER TIMER
VIRTUAL TERMIN
SPI DEBUGGER
I2C DEBUGGER
SIGNAL GENERAT
PATTERN GENER
DC VOLTMETER
DC AMMETER
AC VOLTMETER
AC AMMETER
U1
XTAL1
XTAL2
RST
PSEN
ALE
EA
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
AT89C51
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
LS1
SPEAKER
A
B
C
D
5 Message(s)
ANIMATING: 01:57:54.850000 (CPU load 11%)
+5600.0
-3400.0

IMPLEMENTATION

```
#include<reg51.h>
sbit bu=P2^0;
void delay1on()
{
TMOD=0X01;
TH0=0Xf8;
TL0=0X80;
TR0=1;
while(TF0==0);
TR0=0;
TF0=0;
}
void delay2on()
{
TMOD=0X01;
TH0=0Xf9;
TL0=0X55;
TR0=1;
while(TF0==0);
TR0=0;
TF0=0;
}
void delay3on()
{
TMOD=0X01;
TH0=0Xfa;
```

```
TL0=0X00;
TR0=1;
while(TF0==0);
TR0=0;
TF0=0;
}
void delay4on()
{
TMOD=0X01;
TH0=0Xfa;
TL0=0Xab;
TR0=1;
while(TF0==0);
TR0=0;
TF0=0;
}
void delay5on()
{
TMOD=0X01;
TH0=0XFB;
TL0=0X00;
TR0=1;
while(TF0==0);
TR0=0;
TF0=0;
}
void delay6on()
```

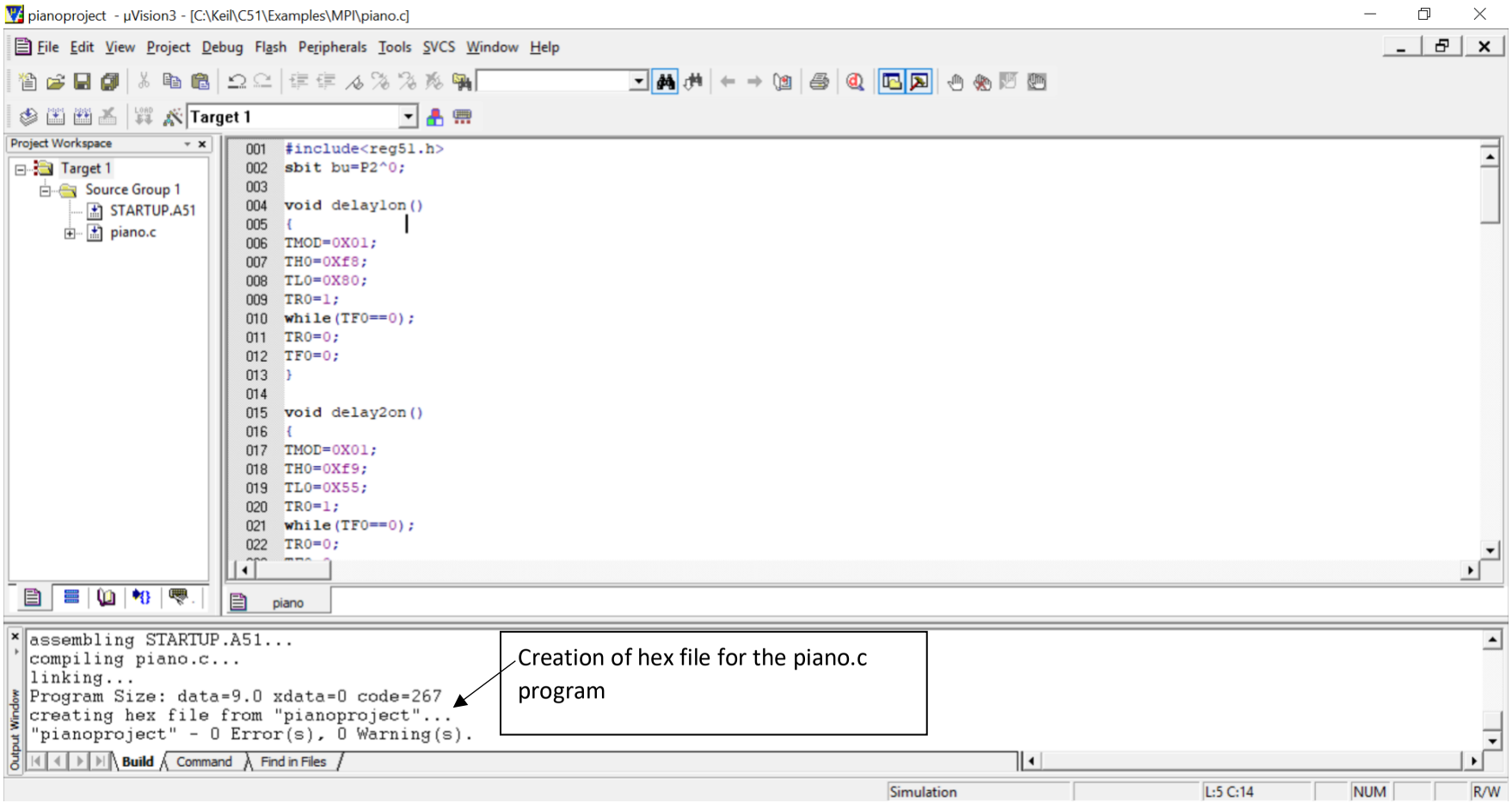
```
{
TMOD=0X01;
TH0=0XFB;
TL0=0X80;
TR0=1;
while(TF0==0);
TR0=0;
TF0=0;
}
void delay7on()
{
TMOD=0X01;
TH0=0XFC;
TL0=0X00;
TR0=1;
while(TF0==0);
TR0=0;
TF0=0;
}
void delay8on()
{
TMOD=0X01;
TH0=0XFC;
TL0=0X40;
TR0=1;
while(TF0==0);
TR0=0;
```

```
TF0=0;
}

void main()
{
P2=0X00;
while(1)
{
if(P1==0xFE)
{
bu=~bu;
delay1on();
}
else if(P1==0xFD)
{
bu=~bu;
delay2on();
}
else if(P1==0XFB)
{
bu=~bu;
delay3on();
}
else if(P1==0XF7)
{
bu=~bu;
delay4on();
```

```
}
else if(P1==0XEF)
{
bu=~bu;
delay5on();
}
else if(P1==0XDF)
{
bu=~bu;
delay6on();
}
else if(P1==0XBF)
{
bu=~bu;
delay7on();
}
else if(P1==0X7F)
{
bu=~bu;
delay8on();
}
}
}
```

Figure 4


pianoproject - µVision3 - [C:\Keil\C51\Examples\MPI\piano.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Source Group 1
STARTUP.A51
piano.c
001 #include<reg51.h>
002 sbit bu=P2^0;
003
004 void delay1on()
005 {
006 TMOD=0X01;
007 TH0=0Xf8;
008 TL0=0X80;
009 TR0=1;
010 while(TF0==0);
011 TR0=0;
012 TF0=0;
013 }
014
015 void delay2on()
016 {
017 TMOD=0X01;
018 TH0=0Xf9;
019 TL0=0X55;
020 TR0=1;
021 while(TF0==0);
022 TR0=0;
piano
assembling STARTUP.A51...
compiling piano.c...
linking...
Program Size: data=9.0 xdata=0 code=267
creating hex file from "pianoproject"...
"pianoproject" - 0 Error(s), 0 Warning(s).
Creation of hex file for the piano.c program
Output Window
Build
Command
Find in Files
Simulation
L:5 C:14
NUM
R/W

According to the buttons pressed, corresponding frequencies are produced. The frequencies are generated in the oscilloscope and can be viewed for further examination of the frequency pulses. And, also before running the simulation the values of frequencies must be same for both the keil project code written as well as proteus simulation. Hence, in the keil target1 that has been generated for current project we need to set "Target-XTAL": 11.0592 MHZ and "Use on chip ROM" are to be selected. Along with this in the "Output" window we need to enable the generate "hex" file check box so that we can add the resultant hex file produced into the Proteus simulation. In the Proteus, after the construction of circuit using the components mentioned, we need to double click on the Atmel microcontroller, and in the window browse the hex file that was generated previously. And donot forget to set up the frequency same as that of in Keil project i.e. 11.0592 MHz.

OUTPUTS in OSCILLOSCOPE:

Different output wave forms produced by the oscilloscope on key press of piano.

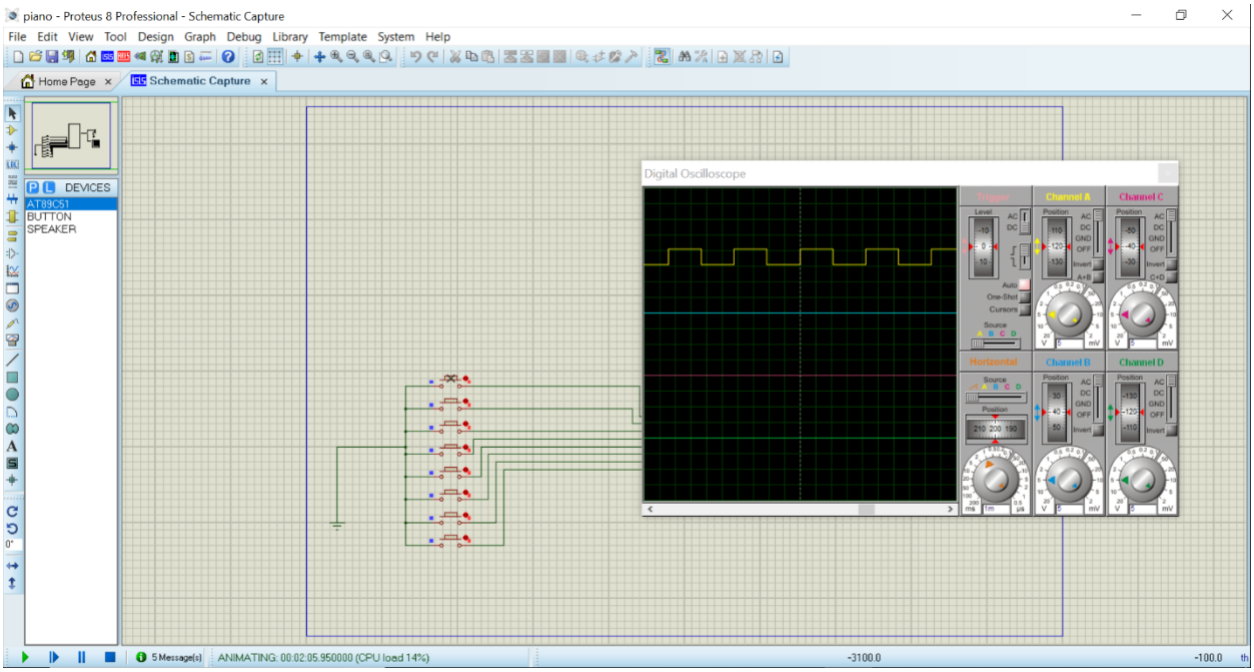


Figure 5 (key 1)

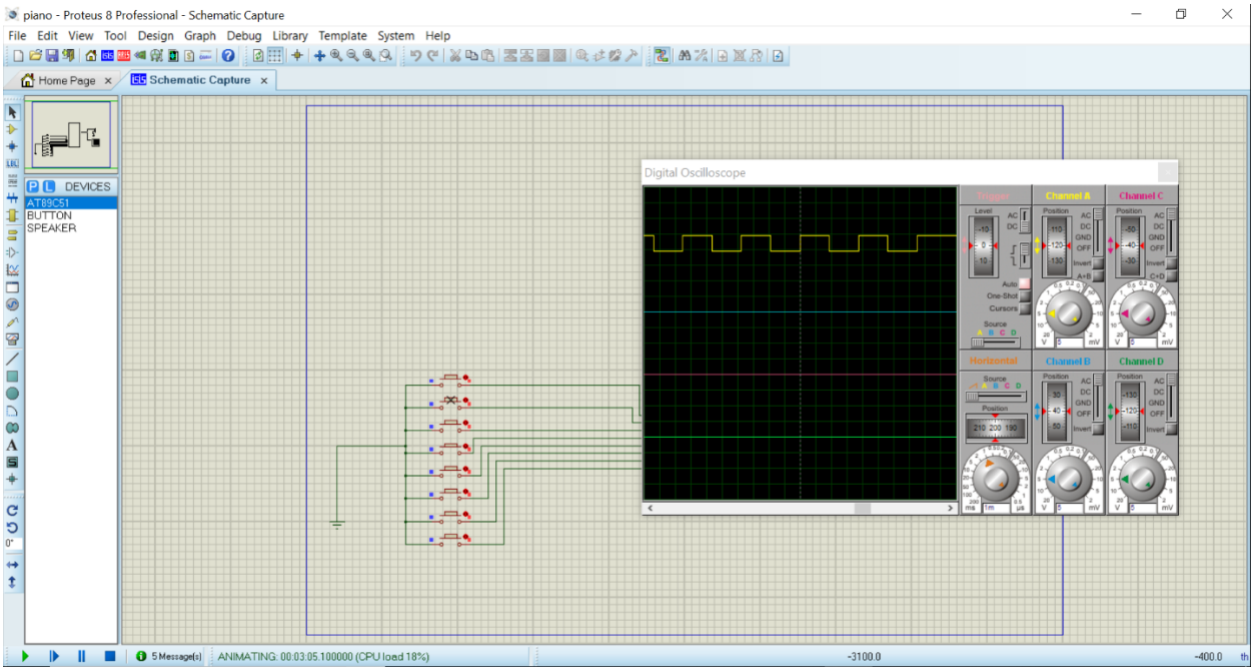


Figure 6 (key 2)

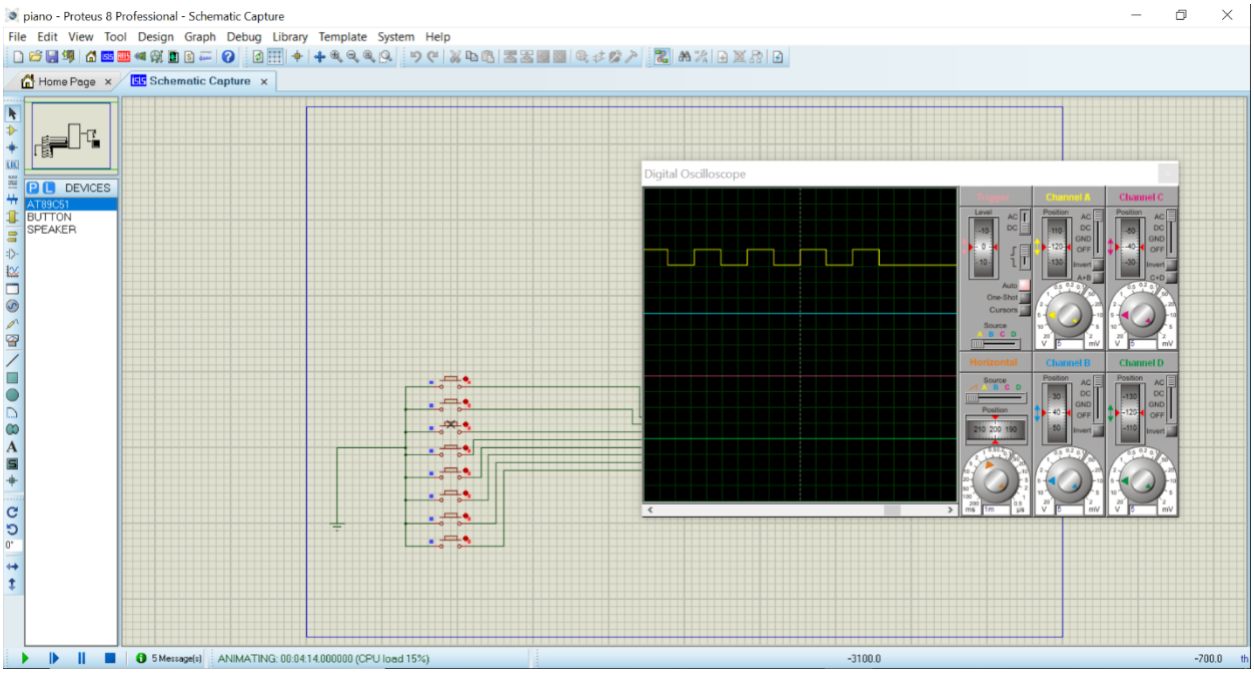


Figure 7 (key 3)

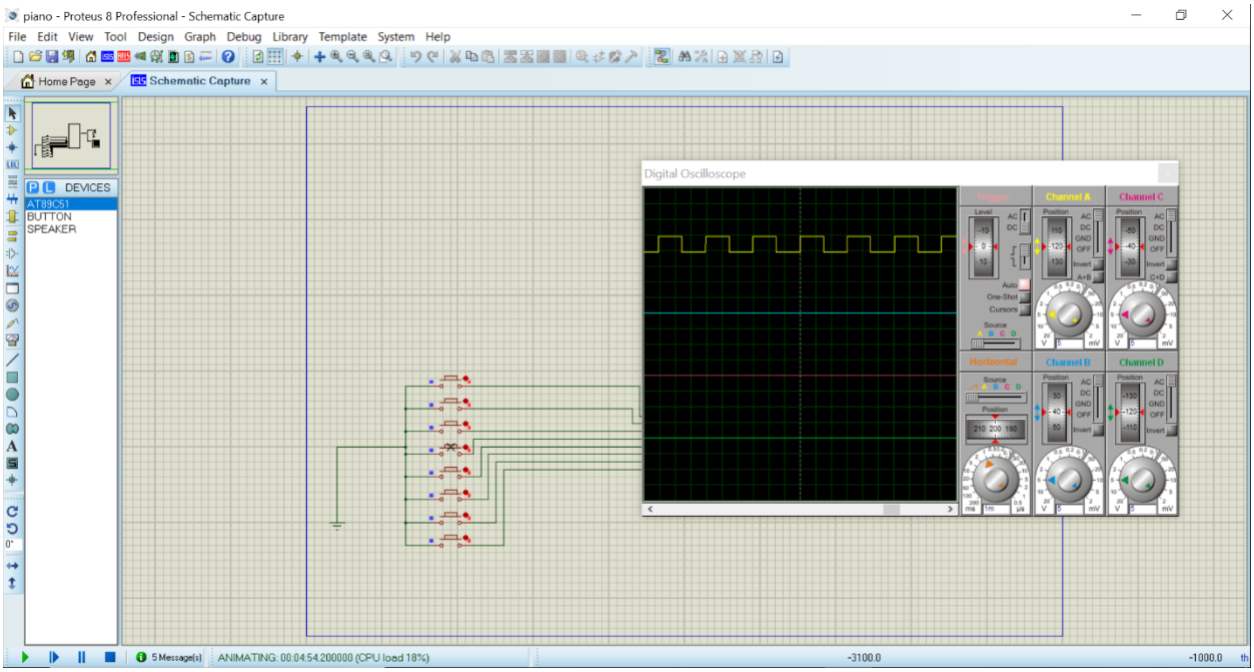


Figure 8 (key 4)

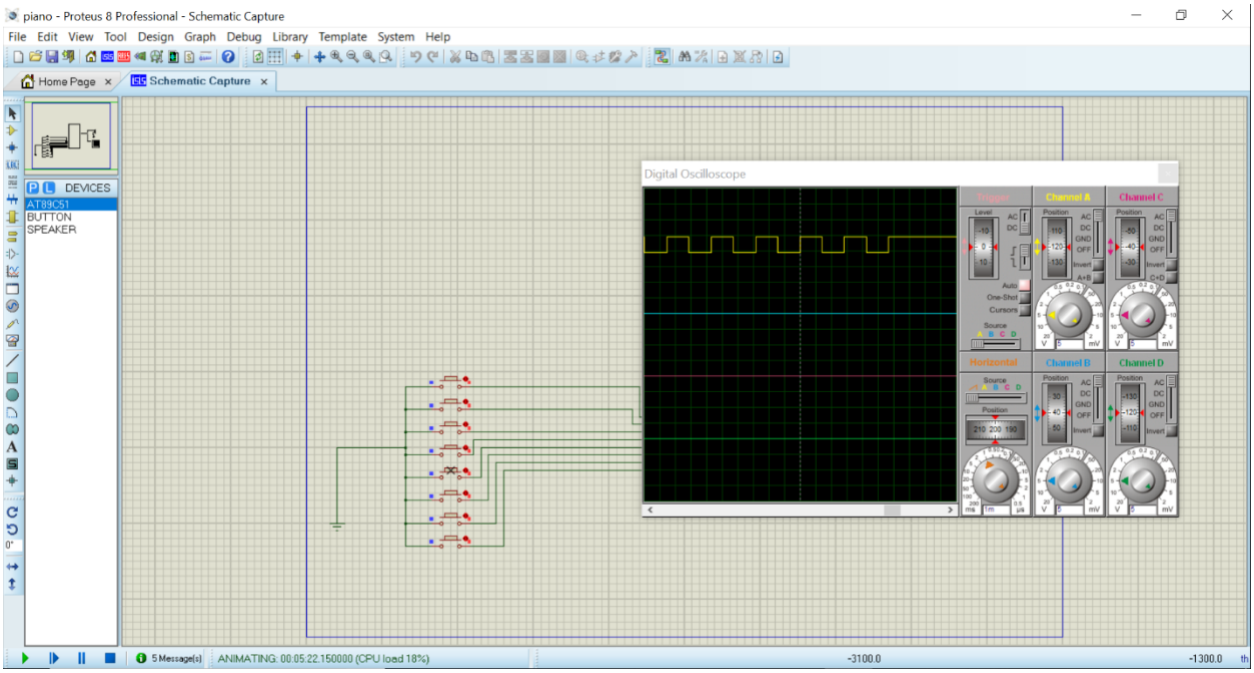


Figure 9 (key 5)

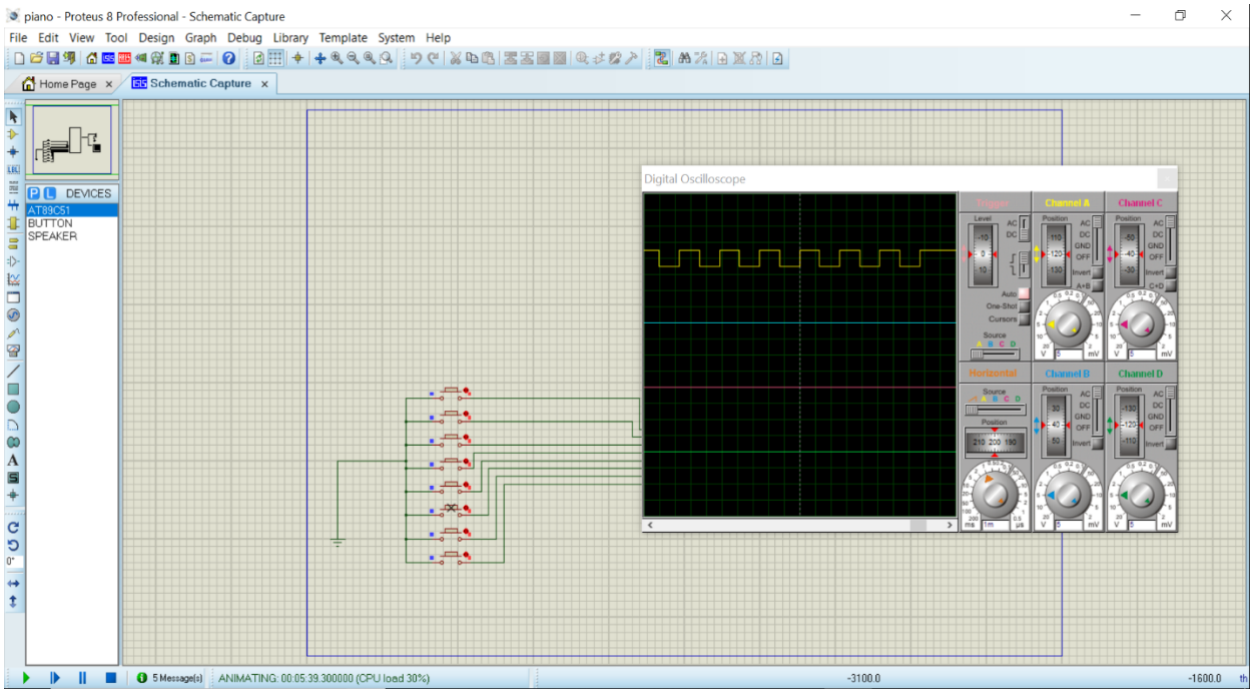


Figure 10 (key 6)

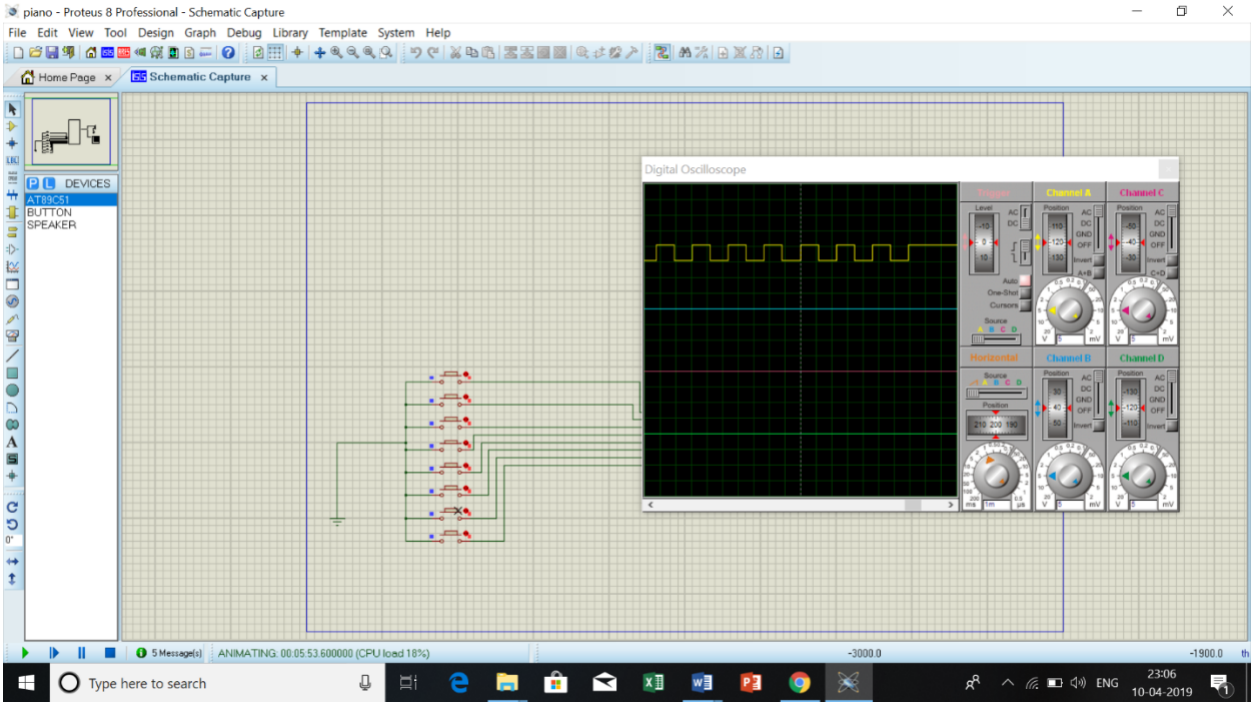


Figure 11 (key 7)

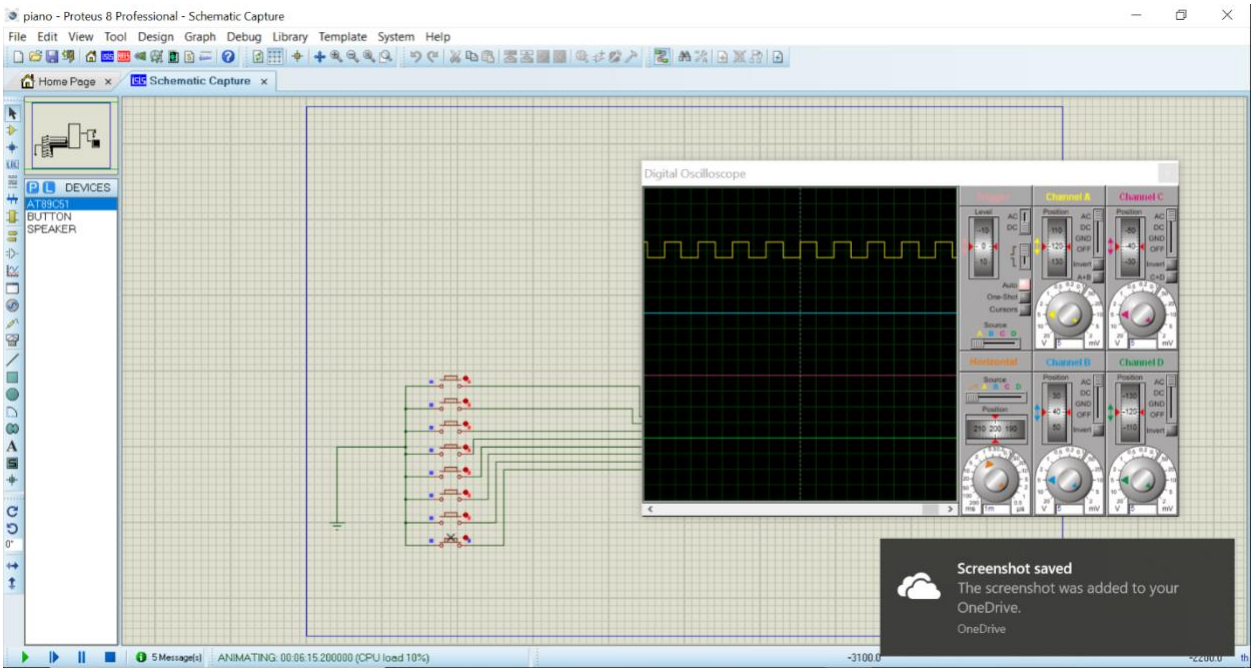


Figure 12 (key 8)

CONCLUSION AND FUTURE WORK:

Finally, with the guidance of our faculty and under the supervision of our respectable H.O.D we are done with our project and I am happy that it is running successfully and I hope that our solution will be useful for all others. Finally, at the end of the day we are done with our project.

I would like to thank our faculty and the department of CSE for giving me this opportunity to learn a lot of things from Keil and Proteus, using which I have developed my project.The project works well as an 8 key piano that produces different frequencies of sound on button press.

REFERENCES:

1. http://www.keil.com/ - KEIL MICROVISION
2. https://electronicswork.wordpress.com/2012/10/19/microcontroller-8051-piano-in-keil-and-proteus-simulator/
3. https://www.youtube.com/watch?v=oWr37z8LIrg
4. https://www.electronicshub.org/8051-microcontroller-introduction/
5. https://www.youtube.com/watch?v=UfTTas8HjVE
6. https://www.labcenter.com/ - PROTEUS