Overview

We have implemented a stopwatch using TM4C123 microcontroller, Buzzer and 7 Segment Display (Common Anode). It uses UART communication to receive the number of minutes the timer should be set. We have used UART0 Module for implementing UART communication as it is connected directly with the PC through the USB cable.

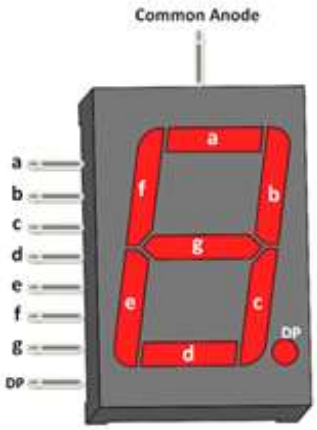
Buzzer

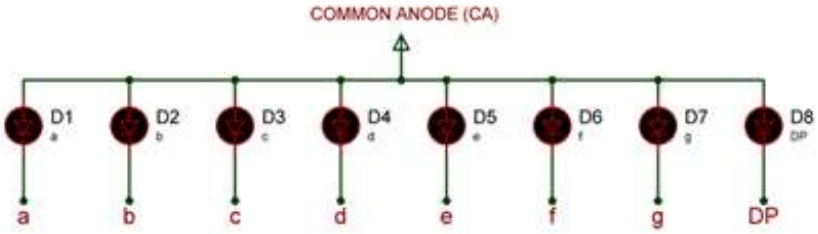
Buzzer is used as an alarm to indicate that the timer has finished running. Any of the two red wires can be connected to Vcc. The black wire must be connected to GND.

7 Segment Display (Common Anode)

7 Segment display is used to display the remaining time from the timer. Each of the individual display segments correspond to different LEDs in the display. Since it's Common Anode, we connect the COM pin to Vcc.

We have three displays in total - one for displaying minutes and two for seconds.





UART Pins

To get the Timer value from the Python GUI running in PC, we need to use a UART communication port of the TM4C123 microcontroller. This microcontroller has on-chip 7 UART ports such as UART0 to UART7. We can use any one of these UART ports to interface TM4C123 with the Python GUI.

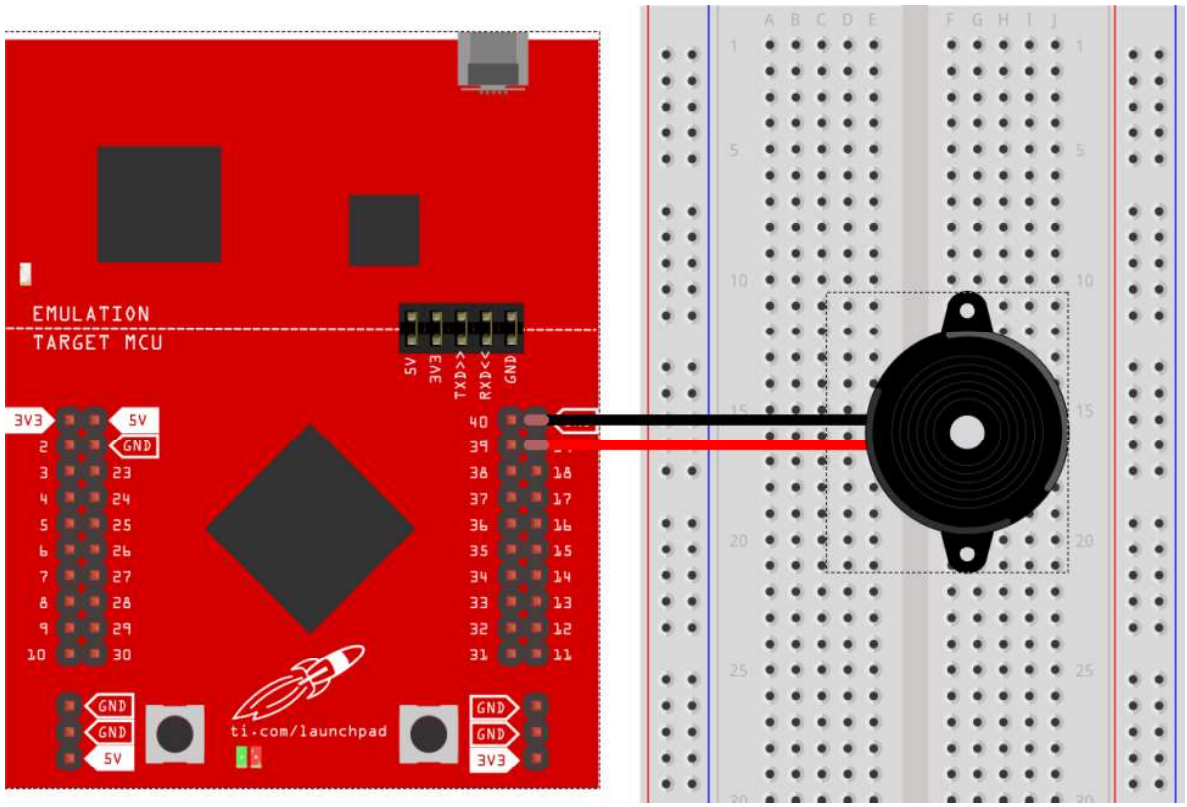
The following table shows all UART ports of the TM4C123 Tiva launchpad and their corresponding GPIO pins.

UART Module	Rx Pin	Tx Pin
UART0	PA0	PA1
UART1	PC4	PC5
UART2	PD6	PD7
UART3	PC6	PC7
UART4	PC4	PC5
UART5	PE4	PE5
UART6	PD4	PD5
UART7	PE0	PE1

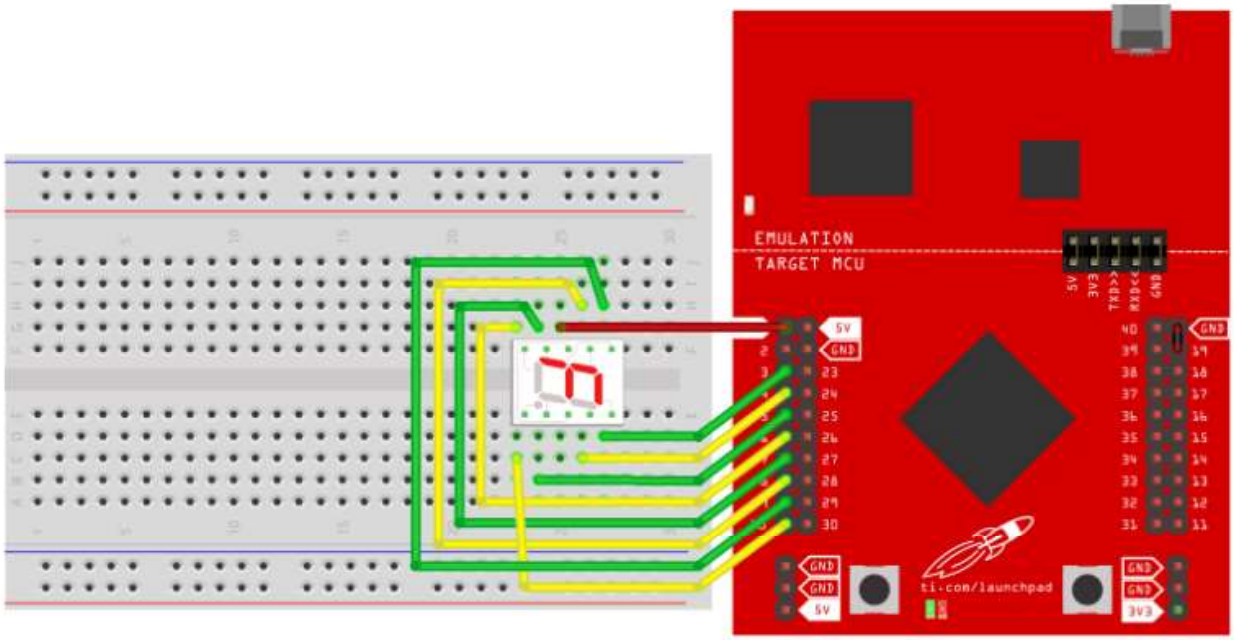
Components Used

1. EK - TM4C123GXL microcontroller
2. Buzzer
3. 7 Segment Display (Common Anode)
4. Breadboard
5. Jumper Wires

Buzzer and TM4C123 Connection Diagram



7 Segment Display and TM4C123 Connection Diagram



GUI

PyQt5 (Python Library)

PyQt is a python binding for Qt (Qt is a cross-platform application development framework for desktop, embedded, and mobile.), which is a set of C++ libraries and development tools that include platform-independent abstractions for graphical user interfaces (GUI).

PyQt5 is a set of Python bindings for **v5** of the Qt application framework and is compatible with Windows, Unix, Linux, macOS, iOS, and Android.

Actions defined in GUI :

- **SET TIME -**

  This button is used to get time as an input from the user. When clicked, this button opens a dialog box where the user can enter the desired time in minutes.

- **START -**

  This button starts the countdown of the timer. The time entered by the user is first sent to the TM4C board as a string via the **Serial** command. When this action is called, it also calls the showTime() method which displays the current time on the GUI. This displayed time decrements by 1 with every passing second.

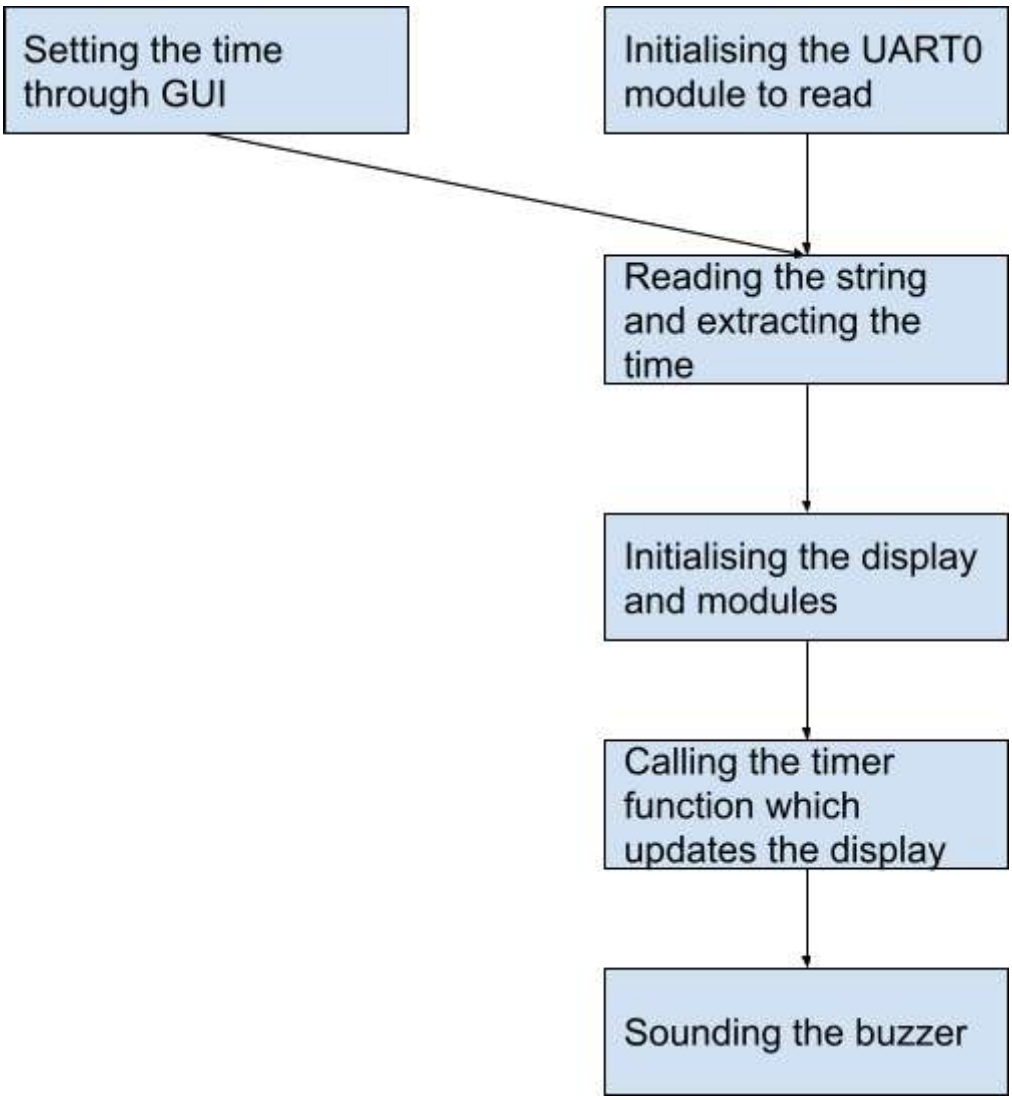
Code flow

First we run the python script that opens up a GUI. The user can enter the number of minutes he/she wishes to set the timer for. Once the timer is set, the user can start the timer by pressing the `START` button. The input given by the user is read as a string by the microcontroller through the UART0 communication module.

The input string is read using `readString()` function which reads input from the UART0 module till the delimiter is read (Here the delimiter is '\r'). Then the minutes are separated from the string and converted as an integer and returned. The returned value is given as input for the `timer()` function which updates the timer by calling `minutes_display()` and `seconds_display()` functions respectively. `minutes_display()` changes the input for the first display and `seconds_display()` changes the input for the remaining two displays.

The values of the 7 segment display corresponding to an integer between 0 and 9 are stored in integer array `numbers`.

NOTE : The register settings for initialising the modules are explained as comments in code given below.



Code

C Code for Stopwatch

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
```

```
#include "tm4c123gh6pm.h"

/* Show numbers in 7 Segment Display (Common Anode) */
int numbers[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80,
0x90};

/* Delay function */
void delayMs(int n)
{
  for(int i = 0; i < n; ++i)
    for(int j = 0; j < 3180; ++j)
      ;
}

/* Read Char by Char from Computer to RX FIFO buffer */
char readChar(void)
{
  char c;

  while((UART0_FR_R & (1 << 4)) != 0);

  c = UART0_DR_R;

  return c;
}

/* Read String from Computer through readchar() function */
char *readString(char delimiter)
{
  int stringSize = 0;
  char *string = (char *)calloc(10, sizeof(char));
  char c = readChar();

  while(c != delimiter)
  {
    *(string + stringSize) = c;
    stringSize++;
    c = readChar();
  }
```

```
  return string;
}

/* Initializing the GPIO ports for UART communication */
void UART_init()
{
  SYSCTL_RCGCUART_R |= (1 << 0); // Enable UART0 Module
  SYSCTL_RCGCGPIO_R |= (1 << 0); // Enable clock to PORTA

  GPIO_PORTA_AFSEL_R = (1 << 1) | (1 << 0); // Enable PA0 and PA1 As
Alternate Function PIN
  GPIO_PORTA_PCTL_R |= (1 << 0) | (1 << 1); // Make PA0, PA1 UART output
pin
  GPIO_PORTA_DEN_R |= (1 << 0) | (1 << 1);  // Enable digital pins PA0, PA1

  UART0_CTL_R &= ~(1 << 0);                  // Disable the UART by clearing
the UARTEN
  UART0_IBRD_R = 104;                       // Integer portion of the BRD
  UART0_FBRD_R = 11;                        // Fractional portion of the BRD
  UART0_LCRH_R = (0x3 << 5) | (1 << 4);        // 8-bit, no parity, 1-stop bit
  UART0_CTL_R = (1 << 0) | (1 << 8) | (1 << 9); // Enable the UART by setting
the UARTEN bit
}

/* Initialize GPIO Ports for displaying minutes in 7 Segment Display */
void minutes_display_init(void)
{
  GPIO_PORTB_CR_R |= 0x7F;        // Allow changes to PB0-PB6
  GPIO_PORTB_AMSEL_R = 0x00;      // Disable analog function
  GPIO_PORTB_PCTL_R = 0x00000000; // GPIO clear bit PCTL
  GPIO_PORTB_DIR_R |= 0x7F;       // Set PB6-PB0 as outputs
  GPIO_PORTB_DEN_R |= 0x7F;       // Enable digital pins PB6-PB0
}

/* Initialize GPIO Ports for displaying seconds in 7 Segment Display */
void seconds_display_init(void)
{
```

```
  /* Second Position 1 - Denotes seconds in multiples of 10 (0, 10, 20, 30, 40,
50) */
  GPIO_PORTE_CR_R |= 0xFF;        // Allow changes to PE0-PE5
  GPIO_PORTE_AMSEL_R = 0x00;      // Disable analog function
  GPIO_PORTE_PCTL_R = 0x00000000; // GPIO clear bit PCTL
  GPIO_PORTE_DIR_R |= 0x3F;       // Set PE0-PE5 as outputs
  GPIO_PORTE_DEN_R |= 0xFF;       // Enable digital pins PE0-PE5

  /* Second Position 2 - Denotes seconds in multiples of 1 (0, 1, 2, 3, 4, 5, 6,
7, 8, 9) */
  GPIO_PORTA_CR_R |= 0xFF;        // Allow changes to PA2-PA7
  GPIO_PORTA_AMSEL_R = 0x00;      // Disable analog function
  GPIO_PORTA_PCTL_R = 0x00000000; // GPIO clear bit PCTL
  GPIO_PORTA_DIR_R |= 0xFF;       // Set PA2-PA7 as outputs
  GPIO_PORTA_DEN_R |= 0xFF;       // Enable digital pins PA2-PA7

  /* PORT D Used in Second Position 1 and 2 */
  GPIO_PORTD_CR_R |= 0xFF;        // Allow changes to PD0, PD7
  GPIO_PORTD_AMSEL_R = 0x00;      // Disable analog function
  GPIO_PORTD_PCTL_R = 0x00000000; // GPIO clear bit PCTL
  GPIO_PORTD_DIR_R |= 0x81;       // Set PD0, PD7 as outputs
  GPIO_PORTD_DEN_R |= 0x81;       // Enable digital pins PD0, PD7
}

/* Initialize GPIO Ports for Buzzer */
void buzzer_init(void)
{
  GPIO_PORTF_DEN_R |= (1 << 4);  // Enable PF4 as digital pin
  GPIO_PORTF_DIR_R |= (1 << 4);  // Configure PF4 as digital output pins
}

/* Get number of minutes from user through UART */
int UART(void)
{
  char *string = readString('\r');
  int time = 0;

  for(int i = 0; (*(string + i) != '\0'); ++i)
  {
```

```
    time = time * 10 + (*(string + i) - '0');
  }

  return time;
}

/* Display minutes remaining in 7 Segment Display */
void minutes_display(int min)
{
  /* Minutes Position - Denotes minutes
  ----------------------------------------
  | TM4C Pin  | Individual Segment |
  |-----------|--------------------|
  |     PB0   |          a         |
  |     PB1   |          b         |
  |     PB2   |          c         |
  |     PB3   |          d         |
  |     PB4   |          e         |
  |     PB5   |          f         |
  |     PB6   |          g         |
  ----------------------------------------
  */

  GPIO_PORTB_DATA_R = numbers[min];
}

/* Display seconds remaining in 7 Segment Display */
void seconds_display(int sec)
{
  /* Second Position 1 - Denotes seconds in multiples of 10 (0, 10, 20, 30, 40,
50)
  ----------------------------------------
  | TM4C Pin  | Individual Segment |
  |-----------|--------------------|
  |     PE0   |          a         |
  |     PE1   |          b         |
  |     PE2   |          c         |
  |     PE3   |          d         |
  |     PE4   |          e         |
```

```
 |     PE5     |             f             |
 |     PD0     |             g             |
 ---------------------------------------
 */

 int s1 = sec / 10;
 int value = (numbers[s1] << 1);
 value = (value >> 1);
 GPIO_PORTE_DATA_R = value;
 GPIO_PORTD_DATA_R = (value >> 6);

 /* Second Position 2 - Denotes seconds in multiples of 1 (0, 1, 2, 3, 4, 5, 6,
7, 8, 9)
 ---------------------------------------
 | TM4C Pin  | Individual Segment |
 |-----------|-----------------------|
 |     PA2     |             a             |
 |     PA3     |             b             |
 |     PA4     |             c             |
 |     PA5     |             d             |
 |     PA6     |             e             |
 |     PA7     |             f             |
 |     PD7     |             g             |
 ---------------------------------------
 */

 int s2 = sec % 10;
 value = (numbers[s2] << 1);
 value = (value >> 1);
 GPIO_PORTA_DATA_R = (value << 2);
 value = (value >> 6);
 GPIO_PORTD_DATA_R |= (value << 7);
}

/* Function for implementing timer */
void timer(int min)
{
 minutes_display(min);    // Display minutes in 7 Segment display
 seconds_display(0);      // Display Seconds in 7 Segment Display
```

```
  for(int i = min - 1; i >= 0; --i)
  {
    for(int j = 59; j >= 0; --j)
    {
      delayMs(1000);      // Delay 1 second (1000 ms = 1 second)
      minutes_display(i); // Display minutes in 7 Segment display
      seconds_display(j); // Display Seconds in 7 Segment Display
    }
  }
}

/* Function to indicate that timer is up (by sound) */
void buzzer(int n)
{
  for(int i = 0; i < n; ++i)
  {
    GPIO_PORTF_DATA_R |= (1 << 4);
    delayMs(100);
    GPIO_PORTF_DATA_R &= 0xEF;
    delayMs(1000);
  }
}

int main()
{
  SYSCTL_RCGCGPIO_R = (1 << 0) | (1 << 1) | (1 << 3) | (1 << 4) | (1 << 5);  //
Enable clock for PORT A, B, D, E, F

  UART_init();
  int count = UART();

  minutes_display_init();
  seconds_display_init();
  buzzer_init();

  timer(count);

  buzzer(5);
```

```
}
```

Python Code for GUI

```
# Importing Libraries
from PyQt5.QtWidgets import *
from PyQt5 import QtCore, QtGui
from PyQt5.QtGui import *
from PyQt5.QtCore import *

import sys
import serial

ser = serial.Serial("COM7", 9600)

class Window(QMainWindow):
    def __init__(self):
        super().__init__()

        # setting title
        self.setWindowTitle("Timer")

        # setting geometry
        self.setGeometry(100, 100, 400, 600)

        # calling method
        self.UiComponents()

        # showing all the widgets
        self.show()

    """ Method for widgets """
    def UiComponents(self):
        # variables
        # count variable
        self.count = 0

        # start flag
        self.start = False
```

```
# creating push button to get time in seconds
button = QPushButton("Set time(mins)", self)

# setting geometry to the push button
button.setGeometry(125, 100, 150, 50)

# adding action to the button
button.clicked.connect(self.get_seconds)

# creating label to show the seconds
self.label = QLabel("Stopwatch", self)

# setting geometry of label
self.label.setGeometry(100, 200, 200, 50)

# setting border to the label
self.label.setStyleSheet("border : 3px solid black")

# setting font to the label
self.label.setFont(QFont('Times', 15))

# setting alignment ot the label
self.label.setAlignment(Qt.AlignCenter)

# creating start button
start_button = QPushButton("Start", self)

# setting geometry to the button
start_button.setGeometry(125, 350, 150, 40)

# adding action to the button
start_button.clicked.connect(self.start_action)

# creating pause button
pause_button = QPushButton("Pause", self)

# setting geometry to the button
pause_button.setGeometry(125, 400, 150, 40)
```

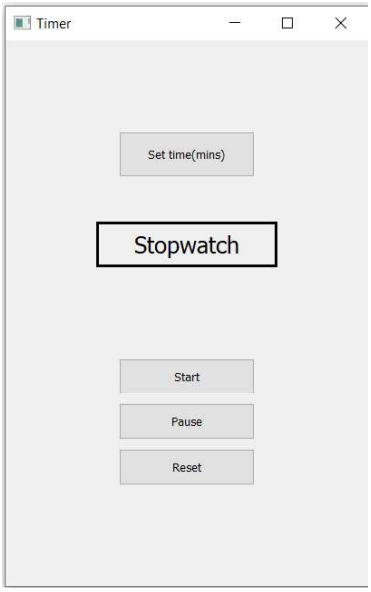
```
    # adding action to the button
    pause_button.clicked.connect(self.pause_action)

    # creating reset button
    reset_button = QPushButton("Reset", self)

    # setting geometry to the button
    reset_button.setGeometry(125, 450, 150, 40)

    # adding action to the button
    reset_button.clicked.connect(self.reset_action)

    # creating a timer object
    timer = QTimer(self)

    # adding action to timer
    timer.timeout.connect(self.showTime)

    # update the timer every tenth second
    timer.start(100)

def send_time(self, time):
    string = str(time) + '\r'
    ser.write(bytes(string, 'utf-8'))

""" Method called by Timer """
def showTime(self):
    # checking if flag is true
    if self.start:
        # decrementing the counter
        self.count -= 1

        # timer is completed
        if self.count == 0:
            # making flag false
            self.start = False

            # setting text to the label
            self.label.setText("Completed !!!! ")
```

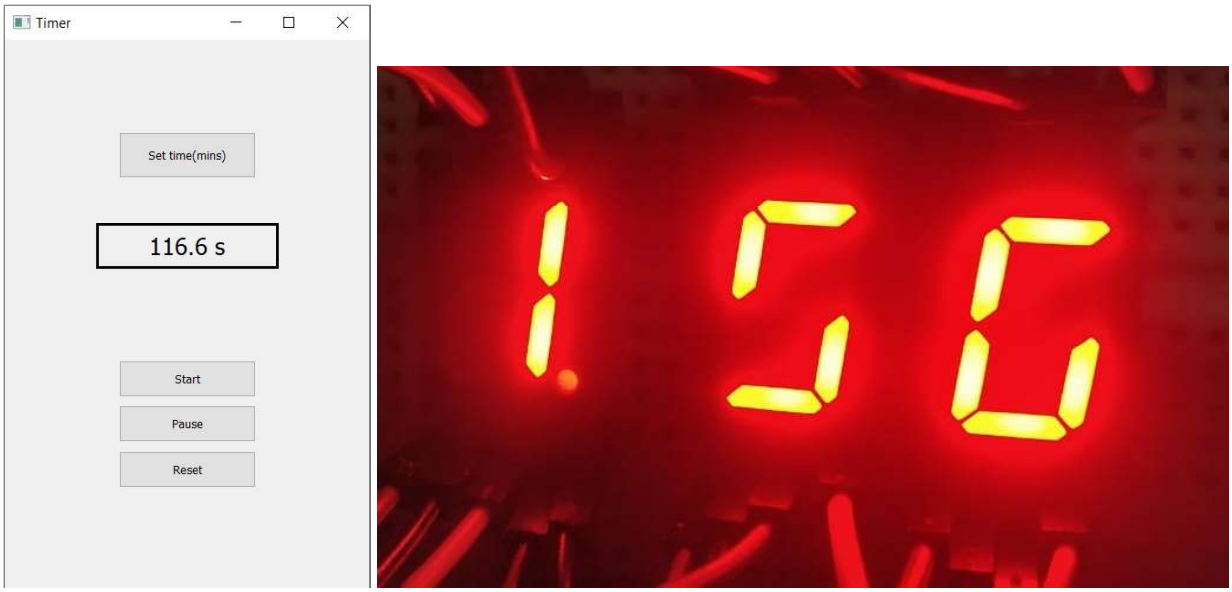
```
    if self.start:
        # getting text from count
        text = str(self.count / 10) + " s"

        # showing text
        self.label.setText(text)

""" Method called by the push button """
def get_seconds(self):
    # making flag false
    self.start = False

    # getting seconds and flag
    mins, done = QInputDialog.getInt(self, 'Minutes', 'Enter Minutes : ')

    # if flag is true
    if done:
        # changing the value of count
        self.count = mins * 600

        # setting text to the label
        self.label.setText(str(mins))

""" Passing value to send_time function """
def start_action(self):
    # making flag true
    mins = self.count//600
    self.send_time(mins)

    self.start = True

    if self.count == 0:
        self.start = False

def pause_action(self):
    # making flag false
    self.start = False
```

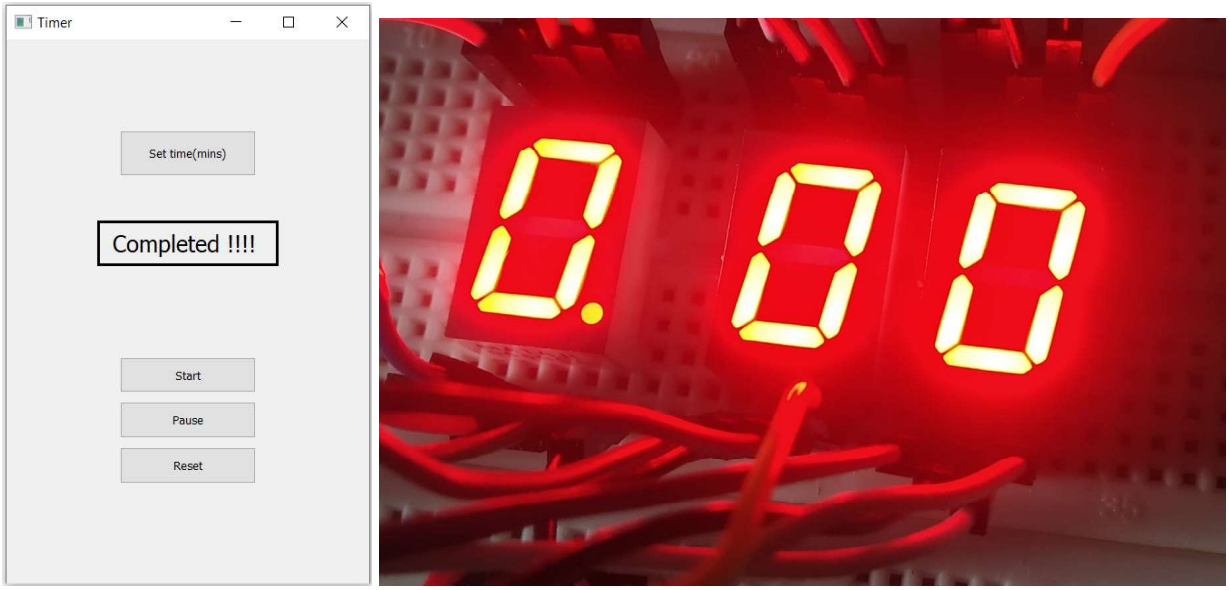
```
    def reset_action(self):
        # making flag false
        self.start = False

        # setting count value to 0
        self.count = 0

        # setting label text
        self.label.setText("Stopwatch")


# create pyqt5 app
App = QApplication(sys.argv)

# create the instance of our Window
window = Window()

# start the app
sys.exit(App.exec())
```

Screenshots and Demo

To see the working, click here.




Timer
Set time(mins)
116.6 s
Start
Pause
Reset


Timer
Set time(mins)
Completed !!!!
Start
Pause
Reset