MAKER: A Braille Clock

Saneela Rabbani, Vaughn College of Aeronautics and Technology

Saneela Rabbani is a senior in Mechatronics Engineering at Vaughn College of Aeronautics and Technology. She is the secretary of the Society of Women Engineers, Vaughn Chapter and secretary of the Unmanned Aerial Vehicle Club. She is a Tutor and Laboratory Technician at Vaughn College. She discovered her passion for teaching at an early age and aspires to obtain her graduate degrees in the field of Engineering and teach on a collegiate level.

Mr. Josiah David D’Arrigo, Vaughn College of Aeronautics and Technology

I am a senior in the Mechatronics Engineering program at Vaughn College of Aeronautics and Technology. I am currently a Tutor at Vaughn College and was secretary of the college’s robotics club for 3 years. My primary engineering interests are aerospace and robotics, with the goal of seeing how they can learn from and contribute to each other to help society advance.

Dr. Shouling He, Vaughn College of Aeronautics and Technology

Dr. Shouling He is an associate professor of Engineering and Technology at Vaughn College of Aeronautics and Technology, where she is teaching the courses in Mechatronics Engineering and Electrical Engineering Technology. Her research interests include modeling and simulation, microprocessors and PLCs, control system designs and Robotics. She has published more than 45 journal and conference papers in these research areas.

Dr. Amir Elzawawy, Vaughn College of Aeronautics and Technology

Dr. Amir Elzawawy teaches courses in Mechanical and Mechatronic Engineering programs. His research background is in the area of experimental fluid mechanics and currently active on the area of CFD (Computational Fluid Dynamics) and heat transfer simulations.

Dr. Hossein Rahemi, Vaughn College of Aeronautics and Technology

Dr. Hossein Rahemi is a professor and department chair of Engineering and Technology at Vaughn College of Aeronautics & Technology. He is the author of two books, Vaughn College Journal of Engineering and Technology (VCJET), numerous conference papers in the areas of solid mechanics, computational mechanics, vibration analysis, fracture mechanics and reliability analysis. He is also a principle investigator for the NSF S-STEM grant and the HIS-STEM grant and a student adviser for a number of technical papers in the areas of mechanics, robotics and industrial automation.

©American Society for Engineering Education, 2016
MAKER: An Innovated Braille Clock

Abstract

Telling time is a fairly simple task for sighted people. However, it poses considerable constraints on the blind and deafblind people and communities. While there are time telling devices available in the market, those designs and implementations are inaccurate and exclusive. Tactile watches allow the user to open the cover of the watch and feel the hour and minutes hand with the corresponding dots raised to indicate the number. This gives the user a general idea of the time. Speaking time telling devices operate by announcing the time when the user presses a button to indicate their need to tell the time. Such devices can go unheard in loud areas and can be problematic and noisy in a quiet classroom/work environment. Additionally, such devices serve no purpose for the deafblind people as they cannot hear. This objective of this project is to develop an inexpensive, user-friendly, digital braille time telling device which will resolve all the inaccuracy and problems existing models pose. This proposal will explore further the details of the working principle, design and implementation of such a device.

Need Statement

For people who are blind, knowing what time it is presents a challenge that non-blind people would hardly think about. Furthermore, deafblind people are presented with an even greater challenge. There are time telling devices such as tactile watches and speaking watches/clock. Such designs can be inaccurate, exclusive and noisy. Additionally, there are little to no digital braille-time telling devices. The objective of this project is to develop an inexpensive, user-friendly, digital braille time telling device.

Background Research

Braille is a writing system created by the Frenchman Louis Braille used by the blind. It uses 6-dot cells and each alphabetic letter is represented through a specific pattern in braille. Due to its universal use, Grade 2 English Braille system and its numerical representations will be used for this project. To represent the numbers 0-9, the braille letters representing the first 10 alphabets are used. Figure 1 displays a representation of Grade 2 English Braille system; at most four dots are needed to represent the numbers 0 through 9.
Design Concepts

In braille, each number is represented as a cell which consists of a variable pattern of four (4) raised dots; to represent the hour and minute, there must be four (4) digits or cells, and each requires four (4) raised dots for a total of sixteen (16) dots. Therefore it was determined that each dot would need to be represented by some physical analog. A manual time set device would closely resemble standard digital watches, where a button is pushed to add time to the hours and minutes. A timer would be programmed to keep track of and update the hours and minutes. Lastly, an external power source will be used to power the clock.

Design Concept 1

The initial design concept called for a total of 1 cells; 4 dots per digits for four digits. A 24 hour format would be used to avoid AM/PM settings. In this design, solenoids serve as the linear actuators which actuate when an electric signal is applied. With 16 solenoids, there is a large current draw of approximately 16 Amperes. This requires a power supply that can supply and support this power requirement. Further, constant use can overdrive the solenoids which would challenge their reliability. Finally, it is an electrical hazard to the user if the clock begins overheating, or if a large amount of current was to pass through their body.
Design Concept 2

The second design concept focused on steering away from the solenoids due to their high current draw. The replacement for the solenoids would be servo motors. Attaching a pin to the motor and then supplying power to the motor will result in the motion of the motor and the pin. Another advantage to the servo motors is that they are precise. However, this design had setbacks as well. Attaching the pins would render this project tedious and prone to errors. The decrease in the current draw was not sufficient. There was still a 10 Ampere current required. Although there are power supplies designed to withstand the current, this design was still too much of a safety hazard. Therefore, a third design concept was developed.

Design Concept 3 – Selected Design

The third design concept uses only four linear actuators, therefore using solenoids seemed to be the most efficient option. Additionally, using micro solenoids that require 3-12 Volts and draw anywhere from 80 to 350 milliAmperes, the current draw lowers substantially. Although there is a small current draw, it is preferred to have the solenoids off until the user wants to check the time. This in turn means that less ports on the microcontroller will be used, thus a smaller microcontroller can be used.
**Working Mechanism of Selected Design (Design Concept 3)**

This design uses a total of four solenoids. Each solenoid represents one cell in braille. The solenoids are placed in a two by two configuration. There are four possible digits that need to be displayed to tell the time. They are the tens digit for hours, the units digit for hours, the tens digit for minutes and the units digit for minutes. This design uses a 24 hour format to eliminate any confusion between AM and PM.

When the clock is first plugged in the clock is automatically set at 00:00. The user must then reset the time, and select and set the desired time. In order to set the time, the user must:

1. Press the “Reset” button.
2. Click the “Select” button to select the hour (0-24). Each click will increment the hour count.
3. Set the hours using the “set” button.
4. Select the tens digit for the minute (0-5) using the “Select” button.
5. Set the tens digit for the minute using the “Set” button.
6. Select the unit digit for the minutes (0-9).
7. Set the units digit for the minute using the “Set” button.
8. Verify the desired time.

Once the desired time has been selected, the microcontroller will begin counting the time from the selected time onwards. When the user wants to know the time, they simply move their finger towards the cells. This area is conductive; it uses conductive tape and a touch sensor to record human interaction. Upon receiving this information, the microcontroller executes the code to actuate the solenoids which display the time. First, the tens digit for hours will actuate for 1 second. Then, all solenoids will retract for 1 second. The same process will occur for the units digit for hours, the tens digit for the minutes and the units digit for the minutes. Once the solenoids retract after displaying units digit for the minutes, the timer continues and all solenoids return to their home position (in this case extended) until new human interaction is detected.

If the user wants to reset the time, they have the option to do so, by following steps one through eight above.
Manufactured Parts

The case for the braille clock was designed in SolidWorks 2015. The design consists of 2 parts: the body and the lid. The entire clock measures 170 mm long, 105 mm wide, and 77 mm tall. The base has an opening in the back for the programming cable to connect to the Arduino, another back opening for power wires, 3 holes for the three buttons on the right side. The buttons are labeled in braille as “Reset”, “Select”, and “Set”, respectively. The base contains all components aside from the buttons. Four pinholes are placed on top for pins from the lid to slide into and connect the two.

The lid of the clock has four circular holes through which the solenoid shafts pass through. Two pins are placed near the edges to go into the pinholes of the base to connect the two.
There is also a holder created for the solenoids. It holds them in the correct orientation within the clock’s body.

Once completed the files were exported to STL files to be printed on a 3D printer. Any type of resin or PLA 3D printer would sufficient for this task.
## Electrical Construction

### Table 1: Parts List

<table>
<thead>
<tr>
<th>Part</th>
<th>Cost (individual)</th>
<th>Cost (total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 x 12 V push solenoid</td>
<td>$7.95</td>
<td>$31.80</td>
</tr>
<tr>
<td>1 x Inland Uno R3</td>
<td>$5.99</td>
<td>$5.99</td>
</tr>
<tr>
<td>1 x 12 V 1 A DC Power Supply</td>
<td>$4.99</td>
<td>$4.99</td>
</tr>
<tr>
<td>1 x Capacitive Touch Sensor</td>
<td>$5.95</td>
<td>$5.95</td>
</tr>
<tr>
<td>2 x Mini Breadboard</td>
<td>$5.99</td>
<td>$11.98</td>
</tr>
<tr>
<td>4 x TIP120 Transistor</td>
<td>$2.50</td>
<td>$2.50</td>
</tr>
<tr>
<td>4 x Tactile Push Button</td>
<td>$5.95</td>
<td>$5.95</td>
</tr>
<tr>
<td>8 x Resistor (various)</td>
<td>$2.00</td>
<td>$2.00</td>
</tr>
<tr>
<td>4 x 1N4001 Diode</td>
<td>$1.50</td>
<td>$1.50</td>
</tr>
<tr>
<td>1 x female 21 mm DC jack</td>
<td>$2.00</td>
<td>$2.00</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>74.66</td>
</tr>
</tbody>
</table>

*Figure 6: Some of the electrical components used to build*
The entire circuit is powered by a single 12 volt 1 amp power supply, connected to a female 21 mm power jack header. An Arduino Uno controls the clock. It was chosen because it had enough pins for all the necessary inputs and outputs and also was capable of taking 12 volts. The transistors and diodes are used to act similar to switches, when a pulse is sent from the Arduino they allow the solenoid to be powered while protecting the board from the current.

The solenoids operate between 3 and 12 volts; the optimal operating voltage for this project is between 4.5 and 5.5 V. The tip of each will act as one dot in the four dot braille-analog matrix. The pattern of raised and lowered solenoids will be used to represent the digits of the current time in braille.

The solenoids themselves will only be activated when the touch sensor detects an input. The touch sensor is capacitive and momentary, only giving a signal when touched. The signal
will tell the board to switch on the correct solenoids in the correct order. The sensor is connected to ground, the board’s 5 V power source, and an input pin on the board.

There are four instances of two different sub-circuits: the solenoid sub-circuit and the push-button sub-circuit. The solenoid sub-circuit consists of the transistor, a diode, a resistor, and solenoid. The emitters share a common ground with the power supply and Arduino. A resistor connects the base to an output pin on the Arduino and the diode is in parallel from the collector with the solenoid.

Figure 8: Solenoid sub-circuit

The push-button sub-circuit is simpler. Each button is directly connected to the board’s 5 V pin through one header. Another header of the button is connected to a voltage divider consisting of a resistor-to-ground line in parallel with a line to an input pin on the board.

Figure 9: Push-button sub-circuit
Software Design

The Arduino software\textsuperscript{6} was used to write and download the program on to the Inland Uno. The code can be found in the Appendix. The Braille Clock requires several functions that a normal clock does not; such as linear actuation. Thus, it was necessary to utilize a microcontroller.

The code is broken down in to several parts. Firstly, libraries are included to access information and functions in the code that are pre-defined and exist already. One of the main libraries used is the TimerOne library which is used to initialize Timer1 and set a 1 second period. This is imperative so that each increment of the seconds in the algorithm matches an actual second.

Next, the variables are declared and initialized. Initialization is important because it ensures that the variables are not assigned random values. There are various functions; TimeCount increments the time, DisplayTime actuates the solenoids and TimeSelection allows the user to input desired time.

When the user powers the clock on, the very first task required of them is to reset the time so that they may set it to a desired time. In order to do that, the user must press the “Reset” button, located on the right face of the clock. When the reset button is pressed, the Uno receives an input of 1. The algorithm is designed in a manner such that, whenever the reset button sends the Uno an input of 1 to a pin, the timer is interrupted. Then, the user is prompted to select the hours desired, using the “Select” button. The user presses the select button till the hours desired are selected. The program reads each input of the select button and stores it in a variable, which then is assigned to the hour position. Once the desired hour is reached, the user presses the “Set” button. Then, the user is prompted to repeat this task for the tens digit and unit digit of the minutes. Once the set button has been pressed 3 times, the timer begins incrementing using the TimeCount function.

Once the time is set, the Uno constantly looks out for two inputs: touch sensor and reset button. If the capacitive touch sensor is touched, it sends an input signal to the microcontroller, which then takes the current time reading, and actuates the solenoids by sending a signal through the pin to the solenoids as an output, which the solenoid receive as input. If the reset button is
pressed, then the button sends an input signal to the Uno through the pin, prompting the user to input the desired time. Below is the flowchart that represents the basic workings of the program.

![Flowchart](image)

---

**Figure 10: Software Development Program Flow Chart**

**Conclusion**

This Braille Clock serves as a proof of concept of a digital braille watch. For those who are visually impaired, telling time can be a challenging task. Thus, it is imperative to create technology which aims to simplify the user’s life. This braille clock is an excellent assistive device which can be used by the blind and the deafblind to simplify the task of time telling. This device is affordable, self-updating and reliable. In addition, it is very silent, unlike other braille time telling devices which may not be suitable in all scenarios. It also gives the user the freedom to reset the time as pleased.

A future implementation to this device is the additions of more solenoids, if there are less current consuming solenoid or linear actuators available. By doing so, the user can learn the time in a shorter time period and with further simplification.
Appendix

#include <Time.h>
#include <TimeLib.h>

// Time array [element 0 stores hours, element 1 stores minutes and element 2 stores seconds]
int Time[3] = {0, 0, 0} ;

//arduino pin assignment variables
int Solenoid1 = 11; // the first Solenoid is connected to the arduino at pin 11
int Solenoid2 = 10; // the second Solenoid is connected to the arduino at pin 10
int Solenoid3 = 9; // the third Solenoid is connected to the arduino at pin 9
int Solenoid4 = 8; // the fourth Solenoid is connected to the arduino at pin 8
int TouchSensor = 7; // the touch sensor is connected to the arduino at pin 7
int ResetButton = 6; // the button used to reset the time is connect to the arduino at pin 6
int TimeSelectButton = 5; // the button used to select the time is connect to the arduino at pin 5
int SetButton = 4; // the button used to set the time is connect to the arduino at pin 4

int flag = 0; // flag serves as a counter in the ResetTime function

boolean MV1 = LOW; // middle variable used to filter button reading signal
boolean MV2 = LOW; // middle variable used to filter button reading signal
boolean MV3 = LOW; // middle variable used to filter button reading signal
boolean MV4 = LOW; // middle variable used to filter button reading signal
boolean MV5 = LOW; // middle variable used to filter button reading signal
boolean MV6 = LOW; // middle variable used to filter button reading signal

//initializing and declaring variables which determine the state of the buttons (HIGH = on, LOW= off)
boolean TouchSensorState = LOW;
boolean ResetButtonState = LOW;
boolean TimeSelectButtonState = LOW;
boolean SetButtonState = LOW;

// prototypes used to notify program that these programs exist as they are called before they're defined
void MOVE (int CurrentTimeReading);
void Alloff ();
void ResetTime();
void setup() {
    Serial.begin(9600);
    // Initializing Timer1 and set a 1 second period
    Timer1.initialize(1000000);
    // Attach TimerCount() as the timer overflow interrupt
    Timer1.attachInterrupt(TimeCount);
    // declaring pins as their types (output/inputs)
    pinMode(Solenoid1, OUTPUT);
    pinMode(Solenoid2, OUTPUT);
    pinMode(Solenoid3, OUTPUT);
    pinMode(Solenoid4, OUTPUT);
    pinMode(TouchSensor, INPUT);
    pinMode(ResetButton, INPUT);
    pinMode(TimeSelectButton, INPUT);
    pinMode(SetButton, INPUT);
}

void loop() {
    TouchSensorState = digitalRead(TouchSensor); // reads and stores current state of the Touch Sensor
    SetButtonState  = digitalRead(SetButton);   // reads and stores current state of the Set Button
    TimeSelectButtonState  = digitalRead(TimeSelectButton); // reads and stores current state of the Time Select Button
    ResetButtonState  = digitalRead(ResetButton); // reads and stores current state of the Reset Button

    if (TouchSensorState == 1) {
        DisplayTime();
        if (ResetButtonState==1) {
            ResetTime();
        }
    }

    // Time Count is the main algorithm behind the clock's timer; increasing the seconds, minutes and hours respectively
    void TimeCount (){
        Time[2]++; // Increment element 2 (seconds) from array Time from the value stored inside the element
        if (Time[2] >59){ // the if statement evaluates whether the values increasing in element 2(seconds) have reached the number 59
            Time[2] = 0; // resets element 2 (seconds) to 0(0-59 are all the seconds options in time)
        }
        Time[1]++; // Increments element 1 (minute) in array Time from the value stored inside the element
        if (Time[1] >59){ // the if statement evaluates whether the values increasing in element 1(minutes) have reached the number 59
        }
    }
```cpp
Time[1] = 0; // resets element 1 (minutes) to 0 (0-59 are all the minutes options in time)
Time[0]++;} // Increments element 0 (hours) in array Time from the value stored inside the element
if (Time[0] == 24) { // the if statement evaluates whether the values increasing in element 0 (hours) have reached 24
    Time[0] = 0; } // resets element 0 (hours) in array Time (0-23 are all the hours option in 24-hour time format)
for( int i = 0; i <= 2; i++){
    Serial.print(Time[i]);
    Serial.print(".");
}
Serial.println(""); // this marks the end of the TimeCount Function

// DisplayFunction is used to seperate element 1 minutes and element 2 (hours) in array Time into Tens and Units Digits
void DisplayTime (){
    int TensDigit = 0;  // declaring a variable of size int, names TensDigit which is initialized and gets a value of 0
    int UnitsDigit = 0; // declaring a variable of size int, names UnitsDigit which is initialized and gets a value of 0
    // this for loop runs for 2 iterations; when i=0 and i=1; i is a variable of size int, used to iterate the loop
    // it starts with a value of 0 and increases the value it holds after completing one loop, until it reaches the value 2. Then it no longer
    // enters the loop. Instead it skips it.
    for (int i=0 ;i<2 ; i++){
        TensDigit = Time[i]/10; // Dividing by ten assigns the tens digit value of the ith element in array Time to the variable TensDigit
        UnitsDigit= Time[i]%10;  // The mod function takes the remainder after dividing from 10 of the ith element in array Time
        AllOff();
        delay(1000);
        MOVE(TensDigit); // sends the value in TensDigit as an input to the function Move
        delay (1000); // creates a delay of 1000 microseconds (ms) or 1 second (s)
        AllOff(); // executes the function AllOff
        delay(1000); // creates a delay of 1000 microseconds (ms) or 1 second (s)
        MOVE (UnitsDigit); // sends the value in UnitsDigit as an input to the function Move
        delay (1000); // creates a delay of 1000 microseconds (ms) or 1 second (s)
        AllOff(); // executes the function AllOff
    }
}
```
delay (500); // creates a delay of 1000 microseconds (ms) or 1 second (s)
} //this marks the end of the for loop
AllOn();
} // this marks the end of the DisplayFunction

// Function MOVE requires a parameter to be passed to it
void MOVE(int CurrentTimeReading){ // This function takes an input variable, which is the current time reading
    // of either the TensDigit or the UnitsDigit of the hours and/or minutes
    if ( CurrentTimeReading == 0 || CurrentTimeReading == 9) // if the digit being passed is either a 0 or a 9
        {digitalWrite (Solenoid1,HIGH);} // retract Solenoid 1
    else {digitalWrite (Solenoid1,LOW);} // for all other digits, keep Solenoid in home position (actuated)
        if (CurrentTimeReading == 1 || CurrentTimeReading == 2 || CurrentTimeReading == 5 || CurrentTimeReading == 8) // if the digit being passed is either a 1, 2, 3, or 8
            {digitalWrite (Solenoid2,HIGH);} // retract Solenoid 2
        else {digitalWrite (Solenoid2,LOW);} // for all other digits, keep Solenoid in home position (actuated)
        if (CurrentTimeReading == 1 || CurrentTimeReading == 3 || CurrentTimeReading == 4 || CurrentTimeReading == 5) // if the digit being passed is either 1,3,4 or 5
            {digitalWrite (Solenoid3,HIGH);} // retract Solenoid 3
        else {digitalWrite (Solenoid3,LOW);} // for all other digits, keep Solenoid in home position (actuated)
        if ( CurrentTimeReading == 1 || CurrentTimeReading == 2 || CurrentTimeReading == 3 || CurrentTimeReading == 6 || CurrentTimeReading == 9) // if the digit being passed is either a 1,2,3,6 or 9
            {digitalWrite (Solenoid4,HIGH);} // retract Solenoid 4
        else {digitalWrite (Solenoid4,LOW);} // for all other digits, keep Solenoid in home position (actuated)

    //
    } // this marks the end of function MOVE

void AllOn(){
digitalWrite (Solenoid1,LOW);
digitalWrite (Solenoid2,LOW);
digitalWrite (Solenoid3,LOW);
digitalWrite (Solenoid4,LOW);
}
void AllOff(){
digitalWrite (Solenoid1,HIGH);
digitalWrite (Solenoid2,HIGH);
```c
void ResetTime()
{
  flag = 1; // assigns the counter "flag" a value of 1
  Timer1.stop(); // stops the timer
  TimeSelection(); // goes to and executes the TimeSelection Function
} // this marks the end of the ResetTime function

// TimeSelection function allows the user to input the hours, tens digit of
the minutes, and units digit of the minutes to set desired time
void TimeSelection()
{
  Time[0]=0; Time[1]=0; Time[2] = 0; // sets all elements in array Time to
  0
  int tens_mins = 0; // this variable is used to store the tens digit of the
  minutes that the user inputs -- used to display the minutes
  boolean button_press = false; //
  boolean setbutton_press =false; //
  int count =0; // variable used to count the number of times the select
  button is clicked to increase hours and tens and units digit of minutes
  int setcount =0; // variable used to increment the number of times set
  button is clicked to determine if
    //the hours or tens and units digit of minutes are being set
    // execture this loop as long as the counter flag has a value of 1-3 assigned
  to it
    // (flag = 1 : hours, flag = 2 :tens digit of minutes, flag = 3 : units digit of
  minutes)
  while( flag <= 3 )
  {
    while (setcount == 0) {
      // In order to avoid any bouncing from the mechanical switches
      // The state of the set button is read 3 times, with a 1 ms delay
      // This ensures that any noise or bouncing isnt read as an input
      // Only physical clicks are read as an input (known as debouncing)
      MV1= digitalRead(SetButton);
      delay(1);
      MV2=digitalRead(SetButton);
      delay(1);
      MV3=digitalRead(SetButton);
      
      // if all three state readings have a high input (input==1)
      // SetButtonState is assigned the value of 1
      if (MV1==HIGH && MV2==HIGH && MV3 ==HIGH)
      { SetButtonState = HIGH;}
```
while(SetButtonState == HIGH){
    // repeats the filtering process to debounce
    MV1 = digitalRead(SetButton);
    delay(1);
    MV2 = digitalRead(SetButton);
    delay(1);
    MV3 = digitalRead(SetButton);
    // SetButtonState is assigned the value of the AND logical operation
    // of all three readings
    SetButtonState = MV1 && MV2 && MV3;
    setbutton_press = true; // assigns setbutton_press a value true
}

// Filters to debounce TimeSelectButton
MV4 = digitalRead(TimeSelectButton);
 delay(1);
MV5 = digitalRead(TimeSelectButton);
 delay(1);
MV6 = digitalRead(TimeSelectButton);
 // if all three state readings have a high input (input==1)
 // TimeSelectButtonState is assigned the value of 1
 if (MV4 == HIGH && MV5 == HIGH && MV6 == HIGH) {
    TimeSelectButtonState = HIGH;
 }
while(TimeSelectButtonState == HIGH) {
    // repeats the filtering process to debounce
    MV4 = digitalRead(TimeSelectButton);
    delay(1);
    MV5 = digitalRead(TimeSelectButton);
    delay(1);
    MV6 = digitalRead(TimeSelectButton);
    // TimeSelectButtonState is assigned the value of the AND operation
    // of all three readings
    TimeSelectButtonState = MV4 && MV5 && MV6;
    button_press = true;
}
// this if statement is used to increment the count, using the while loop
when TimeSelectButton==1
 // count is incremented each time the SetButton is pushed
 // button_press is assigned false so that the count doesn't increment at
 // random
 /this if statement is used to increment the value of hours and minutes
(tens & units digit)
if(button_press == true) {
    button_press = false;
    count++;
}
when SetButton==1
   //this if statement is used to increment setcount, using the while loop
   //setcount is incremented each time the SetButton is pushed
   //setbutton_press is assigned false so that the count doesn't increment
   //this if statement is used to increment the setcount. Each time set
   count is incremented
   //the selection (hours, tens digit of minutes, units digit of minutes) is
   set and moves on to the next selection type
   if(setbutton_press == true){
       setbutton_press = false;
       setcount++;
   }
   //When flag==1, then the value of count is assigned to the element 0
   (hours) of array Time
   if( flag == 1 ){
       Time[0] = count;
   } //When flag equal 2, the value of count is assigned to the local variable
   tens_mins
   else if( flag == 2 ){
       tens_mins = count;
   } //when flag equals 3, the value of count plus the value of tens_mins
   multiplied by 10 is assigned to element 1(minutes) of array Time
   // By multiplying the tens_mins by 10 will allow it to really be in the
tens digit place
   else if( flag == 3 ){
       Time[1] = tens_mins*10 + count;
   } count =0;setcount=0; // count and setcount are always initialized so
that each selection starts at 0
   flag++; // this increments the flag so that it can decide whether or not
   to run the while loop (till flag > 4)
} // This marks the end of the while loop

    Timer1.initialize(1000000); // resumes timer and ensures each second in
clocks timer is standardized to 1 million microseconds (1s)
} // This marks the end of the TimeSelection function

References


