ALARM CLOCK

Ryan, Brown
Section 004
Email: rb23@email.sc.edu
2/9/22
Department of Mechanical Engineering
University of South Carolina
Columbia, SC 29208

INTRODUCTION

Lab 5, the alarm clock, was a project given to students in the Mechatronics course. In this project the student was tasked with creating a circuit consisting of LEDs to light up sequentially over a given period and stay illuminated until the specified period had completed and the reset button had been pressed. Once the period was over, the circuit was supposed to activate a buzzer that sounded every three seconds and continued to play until the reset button was pressed.

The premise was to create essentially a timer that counted to one minute exactly while also still meeting the parameters of the auxiliary components on the board. There were two parts to this project: the circuit, and the code. Both parts had to be created by the student and presented to the TA to receive credit for the project. Using elements from previous lab activities, the student should be able to create a working circuit that can be used as a one-minute timer. If the code was written correctly, the student could extend this timer to any desired length, making it applicable to many situations.

The potential problems of this project were while the function delay() is in use, no other inputs or outputs can be received or sent, respectively, and creating a buzzer that sounded every three seconds after the time had finished. The student was tasked to be creative to solve these problems.

MATERIALS

A list of materials used for this lab include:

- 5 red LEDs
- 1 green LED
- 1 piezometer
- 1 tactile (push-button) switch
- 6 220 Ohm resistors
- 2 10k Ohm resistor
- Arduino microcontroller
- Breadboard
- Wires of various lengths, depending on preference

THE CIRCUIT

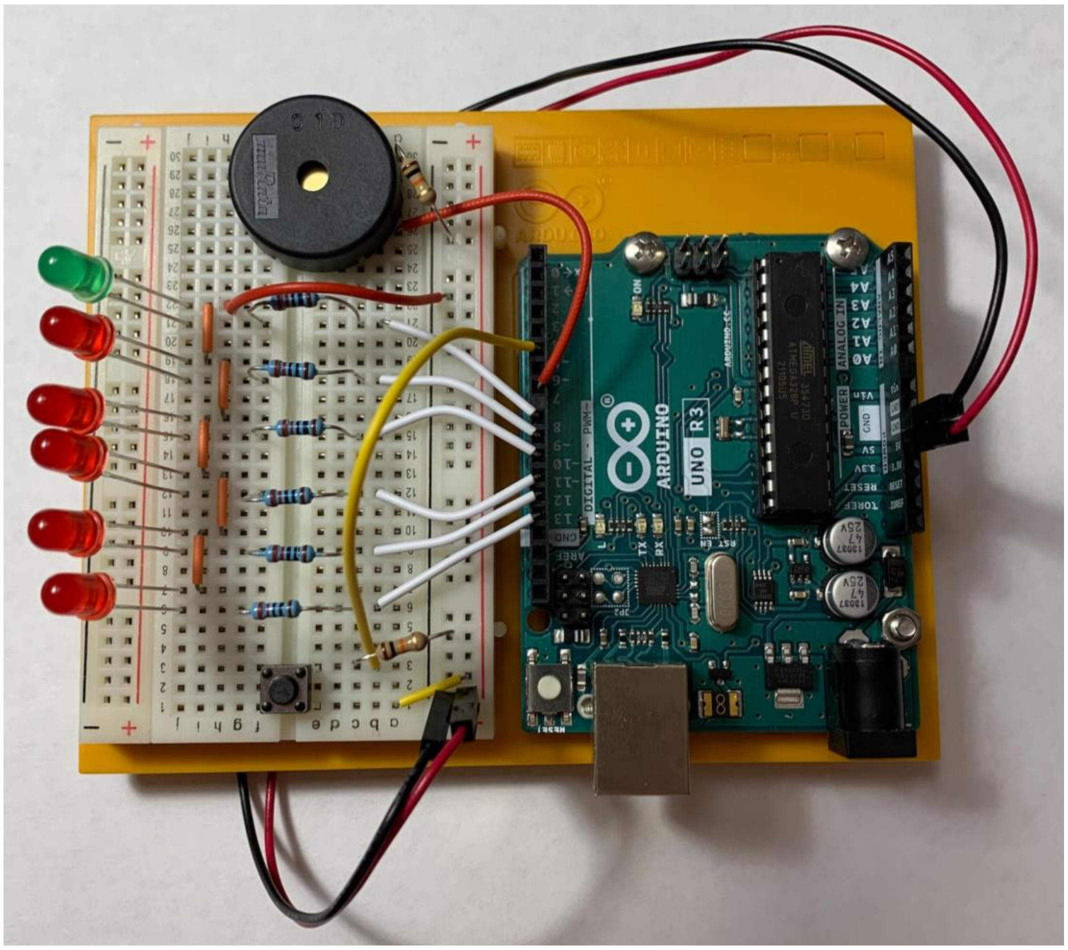


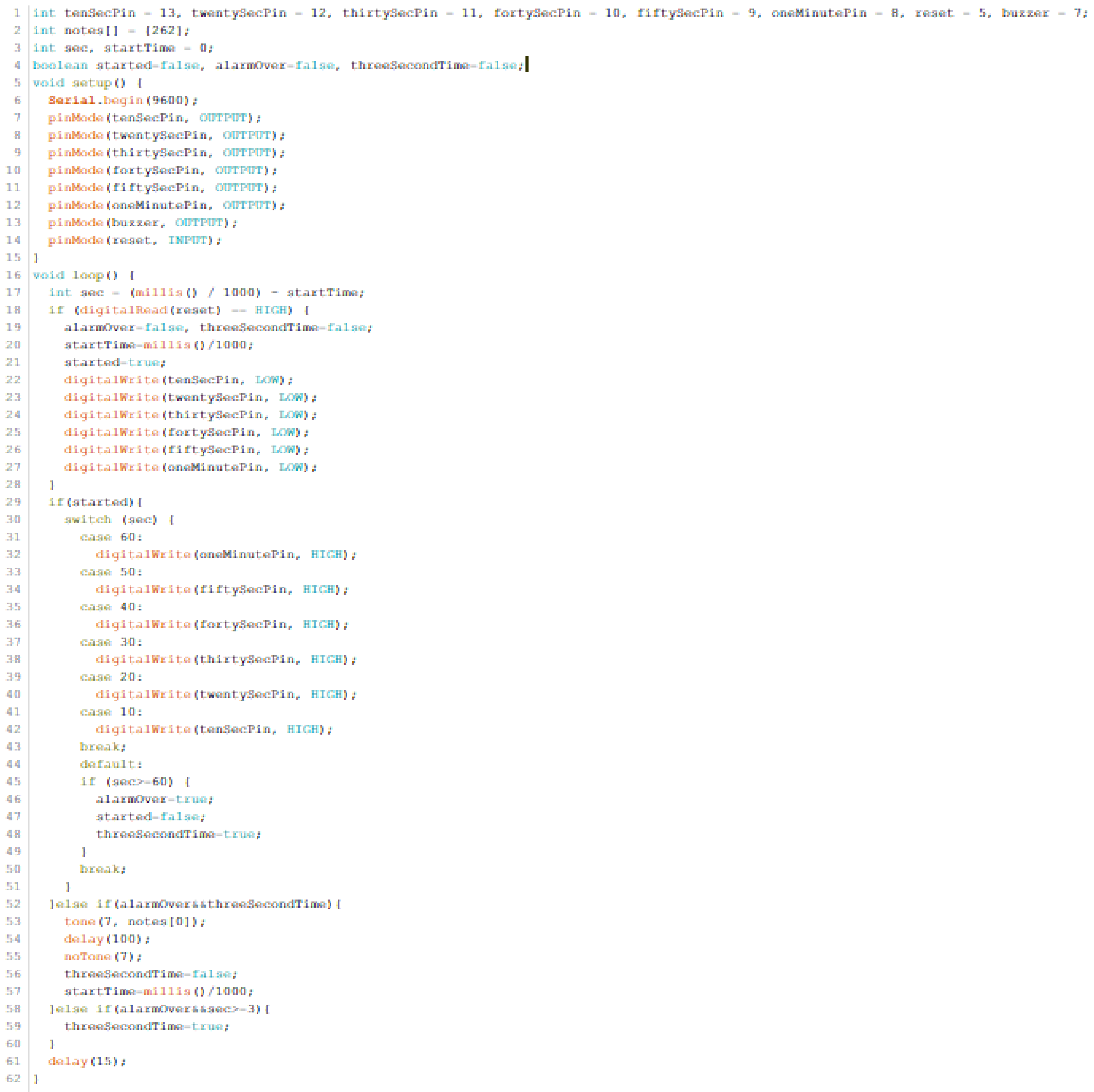
Figure 1: Alarm Clock circuit built on Arduino Uno microcontroller and breadboard

Starting from the very beginning, the red jumper wire is connected to the 5V output pin on the Arduino Uno and connected to the power bus on the breadboard. The black jumper wire is connected to the ground pin on the Arduino Uno and ran to the ground bus on the breadboard. The only component powered by the power bus is the tactile switch, everything else is powered through the digital output pins on the Arduino Uno. The tactile switch is then grounded through a 10k ohm resistor and connected to digital pin 5.

On the left side of Figure 1, six LEDs are shown, five red and one green. To minimize space on the breadboard, all six lead pins have been connected to a common ground and ran to the ground bus on the breadboard. Each LED is connected individually to the Arduino Uno through a 220-ohm resistor and then connected to the digital output pins in descending order from 13 all the way down to 8.

These outputs tell the LEDs to sequentially light up the lights as the code runs. The piezometer is connected to ground through a 10k ohm resistor then powered by digital pin 7. This acts as the buzzer that will play a sound every three seconds after the sixty second time interval has been completed.

THE CODE



```
int tenSecPin = 13, twentySecPin = 12, thirtySecPin = 11, fortySecPin = 10, fiftySecPin = 9, oneMinutePin = 8, reset = 5, buzzer = 7;
int notes[] = {262};
int sec, startTime = 0;
boolean started=false, alarmOver=false, threeSecondTime=false;
void setup() {
  Serial.begin(9600);
  pinMode(tenSecPin, OUTPUT);
  pinMode(twentySecPin, OUTPUT);
  pinMode(thirtySecPin, OUTPUT);
  pinMode(fortySecPin, OUTPUT);
  pinMode(fiftySecPin, OUTPUT);
  pinMode(oneMinutePin, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(reset, INPUT);
}
void loop() {
  int sec = (millis() / 1000) - startTime;
  if (digitalRead(reset) == HIGH) {
    alarmOver=false, threeSecondTime=false;
    startTime=millis()/1000;
    started=true;
    digitalWrite(tenSecPin, LOW);
    digitalWrite(twentySecPin, LOW);
    digitalWrite(thirtySecPin, LOW);
    digitalWrite(fortySecPin, LOW);
    digitalWrite(fiftySecPin, LOW);
    digitalWrite(oneMinutePin, LOW);
  }
  if(started){
    switch (sec) {
      case 60:
        digitalWrite(oneMinutePin, HIGH);
      case 50:
        digitalWrite(fiftySecPin, HIGH);
      case 40:
        digitalWrite(fortySecPin, HIGH);
      case 30:
        digitalWrite(thirtySecPin, HIGH);
      case 20:
        digitalWrite(twentySecPin, HIGH);
      case 10:
        digitalWrite(tenSecPin, HIGH);
      break;
      default:
      if (sec>=60) {
        alarmOver=true;
        started=false;
        threeSecondTime=true;
      }
      break;
    }
  }else if(alarmOver&&threeSecondTime){
    tone(7, notes[0]);
    delay(100);
    noTone(7);
    threeSecondTime=false;
    startTime=millis()/1000;
  }else if(alarmOver&&sec>=3){
    threeSecondTime=true;
  }
  delay(15);
}
```

Figure 2: Code for the Alarm Clock circuit, written in the Arduino app (if difficult to read a physical typed copy of the code will be provided at the end of the report)

The first 4 lines of the code were implemented to set up variables and digital input/output pins. Line 1 assigned the LEDs to individual digital output pins, tenSecPin assigned to 13 all the way to oneMinutePin assigned to pin 8. These pins were responsible for telling the LEDs when to turn on later in the code. Line 2 established the tone that the piezometer would play at the end of the timer. A Boolean system was implemented in assisting the system in telling the timer when to start and reset. In the 'void setup' section of the code, each digital pin was then assigned to be either and input or an output. All variables were assigned to be output except for the reset variable. It was assigned input so that when pressed, the system would start or reset.

Moving to line 17, the 'sec' integer was assigned a function that automatically updates if the button were to be pressed. Line 18 begins the series of 'if' statements that tell the code what to do if an input were to be activated. The system starts at rest and if the button were to be pressed, then the system would stat counting. This was done by the Boolean expressions in line 4, setting everything to be false. Since all three expressions read false, the system does nothing. The if statement in line 18 sets 'alarmOver' and 'threeSecondTime' to be false and 'started' to be true. This was important too the code because it would allow the code to be started and also reset during anytime during the duration of the code. When used in a reset manner this if statement sets all LEDs to low, turning them off, and stops the buzzer. It would also set the 'startTime' integer to be equal to the run time of the program, allowing the sixty second timer to be accurate every time.

Line 29 sets up the second 'if' statement that would be dependent on whether 'started' is equal to true or false. 'Started' equals true once the button is pressed and the code would then run the statement. A 'switch…case' function was implemented for simplicity to the code. The variable the switch statement would run off of would be the integer 'sec'. As the system counts upwards from zero each case would be activated, starting with ten seconds upwards to sixty seconds. Once the sixty second case is reached the switch…case function breaks and moves into thedefault case. The default case has an 'if' statement written into it defined by a value greater that sixty. Once a value greater than sixty is registered it would set 'alarmOver' and 'threeSecond'Time' to true while also setting 'started' to false. After this is done the switch…case function breaks again and moves to the 'else if' statement.

The 'else if' statement depends on whether or not 'alarmOver' and 'threeSecondTime' are both true. If both are true simultaneously then the code will run. The 'tone' function will play the note defined in the argument for a tenth of a second and then turn off. Then the code will set 'threeSecondTime' to false and 'startTime' to equal the current running time, millis(). The next 'else if' tracks if 'alarmOver' is true and if more than three seconds have passed. When both are true it will set 'threeSecondTime' to true and loop back to the previous 'else if' statement.