MEMORANDUM

TO: Dr. Christopher Peters
FROM: Yelnur Abilakim
DATE: 7 Oct 2021
SUBJECT: ECE-303 Lab 2: Timers and Interrupts

Summary

This lab focused on utilizing timers and interrupts to create a code breaking project. The general functionality of the code breaking project is as follows:

- The program generates a random four-digit number between 0000 and 9999.
- Each position of the generated number corresponds to an LED. When the program begins, each LED is blinking. The LEDs are controlled by timers and are initially blinking slowly.
- The user enters a four-digit number. That number is broken down into its four digits, and each digit is compared to the respective integer in the same position as the computer-generated number.
- At each iteration (for each position):
    - If the input number is correct, the corresponding LED turns off.
    - If the input number is incorrect, the corresponding LED blinks faster.
    - After 5 tries, if the user has failed to guess the code, all integers of the number which have not been guessed stay solid.

Four timers were used to control four LEDs.

Introduction

Timer interrupts allow the user to perform a task at very specifically timed intervals regardless of what else is going on in the code. Since the Arduino performs all the commands encapsulated in the loop() function in the order that they are written, it is difficult to time events in the loop(). Arduino timer interrupts allow the user to momentarily pause the normal sequence of events taking place in the loop() function at precisely timed intervals, while executing a separate set of commands. Once these commands are executed, the Arduino picks up right where it was in the loop(). Interrupts are useful for many purposes, including but not limited to: measuring an incoming signal at equally spaced intervals (constant sampling frequency), calculating the time between two events, sending out a signal of a specific frequency, periodically checking for incoming serial data.

Methods

Experimental apparatus used for this lab:

- Breadboard.

- Arduino MEGA 2560 connected to laptop via USB cable.
- Four 1kΩ resistors.
- Four LEDs.
- Five jump wires.

The circuit used for the experiment is shown in Figure 1.

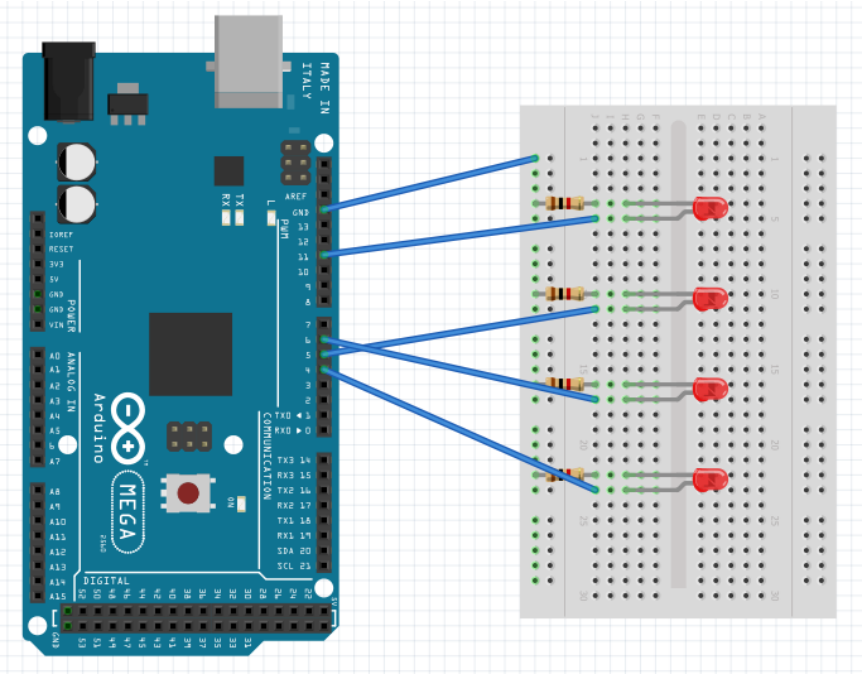


Figure 1. Circuit sketch

One LED is needed for each integer position. Each LED is connected in series with a resistor to limit the LED current (R=1kΩ). Each LED is connected to a digital output pin corresponding to a different timer.

The code for the experiment is appended at the end of the memo.

Results

The project worked as expected, satisfying all functionality requirements described in the Summary section of the memo.

Timers 1, 3, 4, and 5 were chosen to control LEDs 1, 2, 3, and 4, respectively.

Figure 2 displays the physical circuit that was built along with the ID.

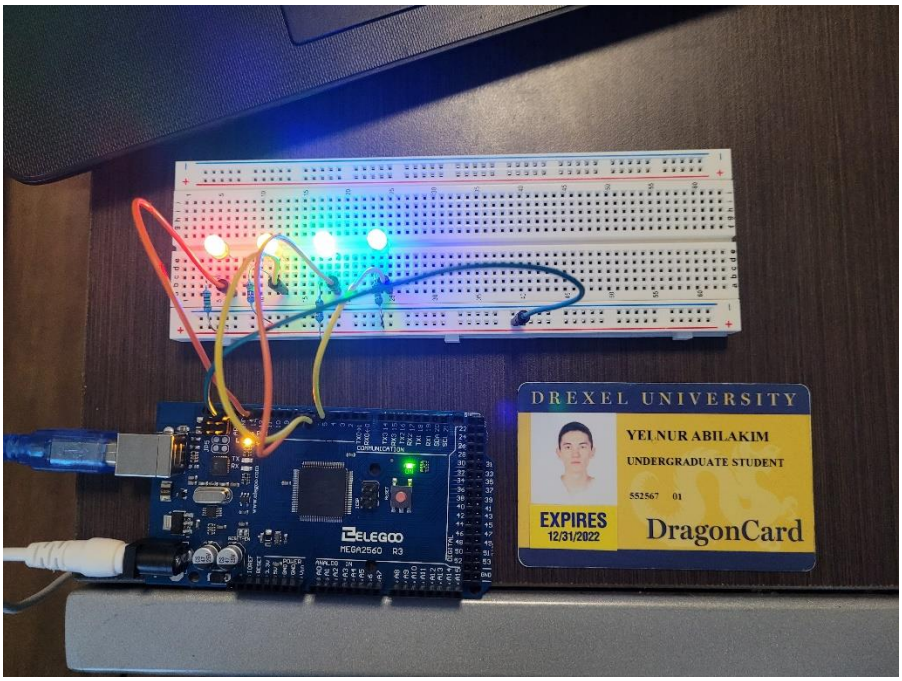


Figure 2. Physical circuit

Conclusions

Timers and interrupts are useful Arduino features that aid in finding solution when there are tasks that need to be performed at specific instances independent of other processes going on in the main loop at the time.

Appendices

Figures 3-7 show the code written for the project.

```
int LED_PIN1 = 11;
int LED_PIN3 = 5;
int LED_PIN4 = 6;
int LED_PIN5 = 4;

String value = "";
long randNumber;
String keys = "";

int set1 = 0;
int set3 = 0;
int set4 = 0;
int set5 = 0;
int tries = 5;

void setup(){
  Serial.begin(9600);
  //set pins as outputs
  pinMode(LED_PIN1, OUTPUT);
  pinMode(LED_PIN3, OUTPUT);
  pinMode(LED_PIN4, OUTPUT);
  pinMode(LED_PIN5, OUTPUT);

  digitalWrite(LED_PIN1, LOW);
  digitalWrite(LED_PIN3, LOW);
  digitalWrite(LED_PIN4, LOW);
  digitalWrite(LED_PIN5, LOW);

  randNumber = random(0, 9999);
  keys = String(randNumber);
  while(keys.length() != 4){
    keys = "0"+keys;
    }
  Serial.print("The key is: ");
  Serial.println(keys);
  noInterrupts();//stop interrupts

  TCCR1A = 0;
  TCCR1B = 0;
```

Figure 3. Code (part 1)

```
  TIMSK1 = 0;
  TCNT1 = 0;
  OCR1A = 30000;
  TCCR1B |= (1 << WGM12);
  TCCR1B |= (1 << CS12) | (0 << CS11) | (0 << CS10);
  TIMSK1 |= (1 << OCIE1A);

  TCCR3A = 0;
  TCCR3B = 0;
  TIMSK3 = 0;
  TCNT3 = 0;
  OCR3A = 30000;
  TCCR3B |= (1 << WGM32);
  TCCR3B |= (1 << CS32) | (0 << CS31) | (0 << CS30);
  TIMSK3 |= (1 << OCIE3A);

  TCCR4A = 0;
  TCCR4B = 0;
  TIMSK4 = 0;
  TCNT4 = 0;
  OCR4A = 30000;
  TCCR4B |= (1 << WGM42);
  TCCR4B |= (1 << CS42) | (0 << CS41) | (0 << CS40);
  TIMSK4 |= (1 << OCIE4A);

  TCCR5A = 0;
  TCCR5B = 0;
  TIMSK5 = 0;
  TCNT5 = 0;
  OCR5A = 30000;
  TCCR5B |= (1 << WGM52);
  TCCR5B |= (1 << CS52) | (0 << CS51) | (0 << CS50);
  TIMSK5 |= (1 << OCIE5A);

  interrupts();//allow interrupts

}//end setup

void loop(){
```

Figure 4. Code (part 2)

```
int flag = 0;
if (Serial.available() > 0){
  // read the incoming byte:
  value = Serial.readString();
  Serial.print("You have entered: ");
  Serial.println(value);
  if (tries > 0){
    if (set1 == 0 && value[0] != keys[0]){
      OCR1A = OCR1A - 5000;
      flag = 1;
    }
    else{
        set1 = 1;
        digitalWrite(LED_PIN1, LOW);
    }
    if (set3 == 0 && value[1] != keys[1]){
        OCR3A = OCR3A - 5000;
        flag = 1;
    }
    else{
        set3 = 1;
        digitalWrite(LED_PIN3, LOW);
    }
    if (set4 == 0 && value[2] != keys[2]){
        OCR4A = OCR4A - 5000;
        flag = 1;
    }
    else{
        set4 = 1;
        digitalWrite(LED_PIN4, LOW);
    }
    if (set5 == 0 && value[3] != keys[3]){
        OCR5A = OCR5A - 5000;
        flag = 1;
    }
    else{
        set5 = 1;
        digitalWrite(LED_PIN5, LOW);
    }
```

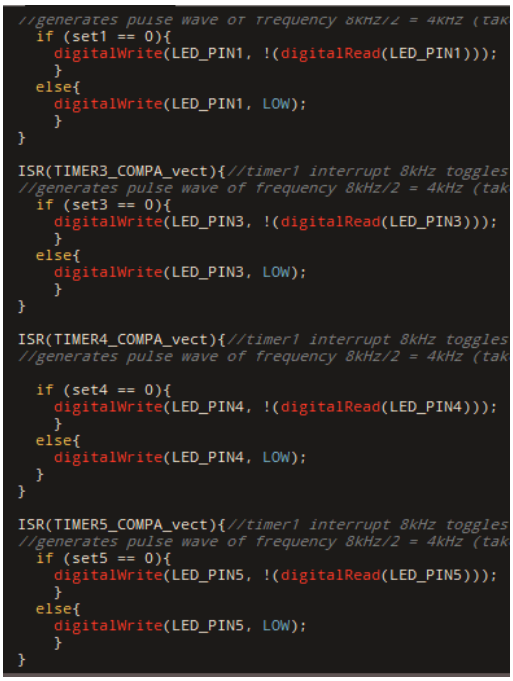
Figure 5. Code (part 3)

```
    if (flag == 1){
      tries = tries - 1;
      if (tries <= 0){
          if (set1 == 0){
            OCR1A = 1;
            digitalWrite(LED_PIN1, HIGH);
            }
          if (set3 == 0){
            OCR3A = 1;
            digitalWrite(LED_PIN3, HIGH);
            }
          if (set4 == 0){
            OCR4A = 1;
            digitalWrite(LED_PIN4, HIGH);
            }
          if (set5 == 0){
            OCR5A = 1;
            digitalWrite(LED_PIN5, HIGH);
            }
      }
    }
  }
 }
}


ISR(TIMER1_COMPA_vect){//timer1 interrupt 8kHz toggles
//generates pulse wave of frequency 8kHz/2 = 4kHz (tak
  if (set1 == 0){
    digitalWrite(LED_PIN1, !(digitalRead(LED_PIN1)));
    }
  else{
    digitalWrite(LED_PIN1, LOW);
    }
}

ISR(TIMER3_COMPA_vect){//timer1 interrupt 8kHz toggles
//generates pulse wave of frequency 8kHz/2 = 4kHz (tak
```

Figure 6. Code (part 4)



```
//generates pulse wave of frequency 8kHz/2 = 4kHz (tak
  if (set1 == 0){
    digitalWrite(LED_PIN1, !(digitalRead(LED_PIN1)));
    }
  else{
    digitalWrite(LED_PIN1, LOW);
    }
}

ISR(TIMER3_COMPA_vect){//timer1 interrupt 8kHz toggles
//generates pulse wave of frequency 8kHz/2 = 4kHz (tak
  if (set3 == 0){
    digitalWrite(LED_PIN3, !(digitalRead(LED_PIN3)));
    }
  else{
    digitalWrite(LED_PIN3, LOW);
    }
}

ISR(TIMER4_COMPA_vect){//timer1 interrupt 8kHz toggles
//generates pulse wave of frequency 8kHz/2 = 4kHz (tak

  if (set4 == 0){
    digitalWrite(LED_PIN4, !(digitalRead(LED_PIN4)));
    }
  else{
    digitalWrite(LED_PIN4, LOW);
  }
}

ISR(TIMER5_COMPA_vect){//timer1 interrupt 8kHz toggles
//generates pulse wave of frequency 8kHz/2 = 4kHz (tak
  if (set5 == 0){
    digitalWrite(LED_PIN5, !(digitalRead(LED_PIN5)));
    }
  else{
    digitalWrite(LED_PIN5, LOW);
    }
}
```

Figure 7. Code (part 5)