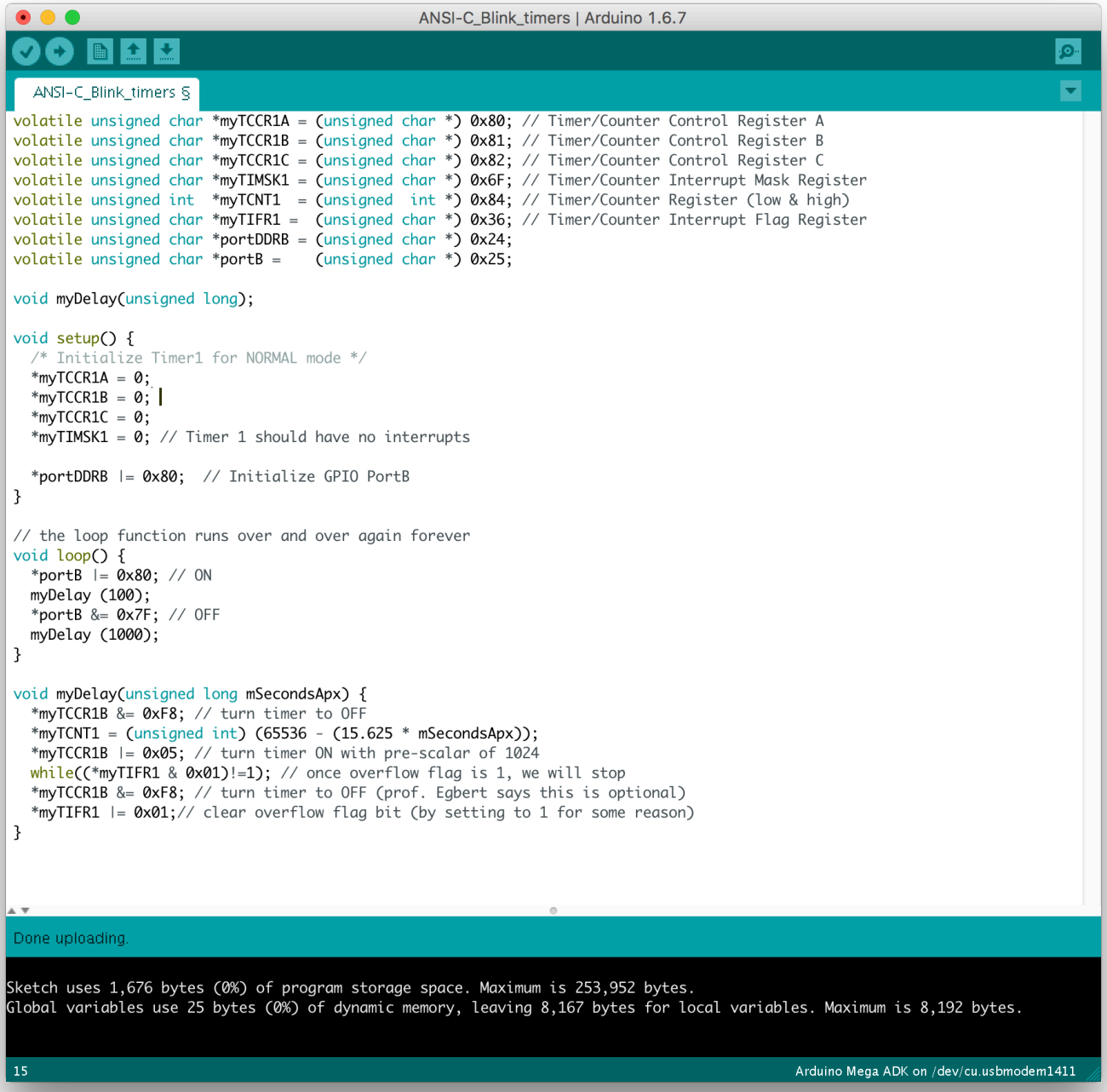
Willis T. Allstead
Professor Egbert
CPE 301
12 October 2016

HW 5

1) Below is a screenshot of the compiled program, all code visible



2) Below is the code for this part, with comments in the myDelay function explaining my math process

ANSI-C_Blink_timers_2 §

```
volatile unsigned char *myTCCR1A = (unsigned char *) 0x80; // Timer/Counter Control Register A
volatile unsigned char *myTCCR1B = (unsigned char *) 0x81; // Timer/Counter Control Register B
volatile unsigned char *myTCCR1C = (unsigned char *) 0x82; // Timer/Counter Control Register C
volatile unsigned char *myTIMSK1 = (unsigned char *) 0x6F; // Timer/Counter Interrupt Mask Register
volatile unsigned int  *myTCNT1  = (unsigned  int *) 0x84; // Timer/Counter Register (low & high)
volatile unsigned char *myTIFR1 =  (unsigned char *) 0x36; // Timer/Counter Interrupt Flag Register
volatile unsigned char *portDDRB = (unsigned char *) 0x24;
volatile unsigned char *portB =    (unsigned char *) 0x25;

void myDelay();

void setup() {
  /* Initialize Timer1 for NORMAL mode */
  *myTCCR1A = 0;
  *myTCCR1B = 0;
  *myTCCR1C = 0;
  *myTIMSK1 = 0; // Timer 1 should have no interrupts

  *portDDRB |= 0x80;  // Initialize GPIO PortB
}

void loop() {
  *portB |= 0x40; // turn on bit 6 (0b01000000 => 0x40)
  myDelay();
  *portB &= 0xBF; // turn off bit 6 (0b10111111 => 0xBF)
  myDelay();
}

void myDelay() {
  /* TODO: generate frequency of 440Hz */

  /*  period => 1/440 = 0.002222222222 seconds = 2.22 milliseconds
   *  half period for square wave => 0.5(1/440) = 1.136363636 milliseconds
   *  we want the delay to be 1.136363636 milliseconds
   *  1.136363636 milliseconds = 1136.363636 microseconds
   *  1136.363636 microseconds/4 microseconds = 284.090909 ~= 284
   *  we need to subtract 284 from the top of the timer
   *
   */
  *myTCCR1B &= 0xF8; // turn timer to OFF
  *myTCNT1 = (unsigned int) (65536 - 284);
  *myTCCR1B |= 0x03; // turn timer ON with pre-scalar of 64 (0b011 => 0x03)
  while((*myTIFR1 & 0x01)!=1); // once overflow flag is 1, we will stop
  *myTCCR1B &= 0xF8; // turn timer to OFF (prof. Egbert says this is optional)
  *myTIFR1 |= 0x01;// clear overflow flag bit (by setting to 1 for some reason)
}
```

Done uploading.

```
Sketch uses 856 bytes (0%) of program storage space. Maximum is 253,952 bytes.
Global variables use 25 bytes (0%) of dynamic memory, leaving 8,167 bytes for local variables. Maximum is 8,192 bytes.
```

3) Below is the code for this part, with comments in the myDelay function explaining my math process

ANSI-C_Blink_timers_3

```
volatile unsigned char *myTCCR1A = (unsigned char *) 0x80; // Timer/Counter Control Register A
volatile unsigned char *myTCCR1B = (unsigned char *) 0x81; // Timer/Counter Control Register B
volatile unsigned char *myTCCR1C = (unsigned char *) 0x82; // Timer/Counter Control Register C
volatile unsigned char *myTIMSK1 = (unsigned char *) 0x6F; // Timer/Counter Interrupt Mask Register
volatile unsigned int  *myTCNT1  = (unsigned  int *) 0x84; // Timer/Counter Register (low & high)
volatile unsigned char *myTIFR1 =  (unsigned char *) 0x36; // Timer/Counter Interrupt Flag Register
volatile unsigned char *portDDRB = (unsigned char *) 0x24;
volatile unsigned char *portB =    (unsigned char *) 0x25;

void myDelay();

void setup() {
  /* Initialize Timer1 for NORMAL mode */
  *myTCCR1A = 0;
  *myTCCR1B = 0;
  *myTCCR1C = 0;
  *myTIMSK1 = 0; // Timer 1 should have no interrupts

  *portDDRB |= 0x80;  // Initialize GPIO PortB
}

void loop() {
  *portB |= 0x80; // turn on bit 6 (0b01000000 => 0x40)
  myDelay();
  *portB &= 0x7F; // turn off bit 6 (0b10111111 => 0xBF)
  myDelay();
}

void myDelay() {
  /* TODO: generate frequency of 12 kHz = 12000Hz */

  /*  period => 1/12000 = 0.00008333333333 seconds = 0.08333 milliseconds
   *  half period for square wave => 0.5(1/12000) = 0.04166666667 milliseconds
   *  we want the delay to be 0.04166666667 milliseconds
   *  0.04166666667 milliseconds = 41.66666667 microseconds
   *  41.66666667 microseconds/0.0625 microseconds = 666.66666672 ~= 667
   *  we need to subtract 667 from the top of the timer and use pre-scalar of 1
   *
   */
  *myTCCR1B &= 0xF8; // turn timer to OFF
  *myTCNT1 = (unsigned int) (65536 - 667);
  *myTCCR1B |= 0x01; // turn timer ON with pre-scalar of 1 (0b001 => 0x01)
  while((*myTIFR1 & 0x01)!=1); // once overflow flag is 1, we will stop
  *myTCCR1B &= 0xF8; // turn timer to OFF (prof. Egbert says this is optional)
  *myTIFR1 |= 0x01;// clear overflow flag bit (by setting to 1 for some reason)
}
```

Done compiling.

```
Sketch uses 856 bytes (0%) of program storage space. Maximum is 253,952 bytes.
Global variables use 25 bytes (0%) of dynamic memory, leaving 8,167 bytes for local variables. Maximum is 8,192 bytes.
```

1

4) Below is the code for this part, with comments in the genFreq function explaining my math process

```
volatile unsigned char *myTCCR1A = (unsigned char *) 0x80; // Timer/Counter Control Register A
volatile unsigned char *myTCCR1B = (unsigned char *) 0x81; // Timer/Counter Control Register B
volatile unsigned char *myTCCR1C = (unsigned char *) 0x82; // Timer/Counter Control Register C
volatile unsigned char *myTIMSK1 = (unsigned char *) 0x6F; // Timer/Counter Interrupt Mask Register
volatile unsigned int  *myTCNT1  = (unsigned  int *) 0x84; // Timer/Counter Register (low & high)
volatile unsigned char *myTIFR1 =  (unsigned char *) 0x36; // Timer/Counter Interrupt Flag Register
volatile unsigned char *portDDRB = (unsigned char *) 0x24;
volatile unsigned char *portB =    (unsigned char *) 0x25;

void genFreq();

void setup() {
  /* Initialize Timer1 for NORMAL mode */
  *myTCCR1A = 0;
  *myTCCR1B = 0;
  *myTCCR1C = 0;
  *myTIMSK1 = 0; // Timer 1 should have no interrupts

  *portDDRB |= 0x80;  // Initialize GPIO PortB
}

void loop() {
  genFreq();
}

void genFreq() {
  /* TODO: generate frequency of 500 Hz, 30% up <---> 70% down */

  /*  period => 1/500 = 0.002 seconds = 2 milliseconds
   *  30% period for square wave => 0.3(1/500) = 0.6 milliseconds
   *  we want the delay to be 0.6 milliseconds for UP
   *  0.6 milliseconds = 600 microseconds
   *  600 microseconds/4 microseconds = 150
   *  we need to subtract 150 from the top of the timer and use pre-scalar of 64
   */

  *portB |= 0x40; // turn on bit 6 (0b01000000 => 0x40) // turn on bit 6

  *myTCCR1B &= 0xF8; // turn timer to OFF
  *myTCNT1 = (unsigned int) (65536 - 150);
  *myTCCR1B |= 0x03; // turn timer ON with pre-scalar of 64 (0b011 => 0x03)
  while((*myTIFR1 & 0x01)!=1); // once overflow flag is 1, we will stop
  *myTCCR1B &= 0xF8; // turn timer to OFF (prof. Egbert says this is optional)
  *myTIFR1 |= 0x01;// clear overflow flag bit (by setting to 1 for some reason)

 /*  period => 1/500 = 0.002 seconds = 2 milliseconds
  *  70% period for square wave => 0.7(1/500) = 1.4 milliseconds
  *  we want the delay to be  1.4 milliseconds for DOWN
  *  1.4 milliseconds = 1400 microseconds
  *  1400 microseconds/4 microseconds = 350
  *  we need to subtract 350 from the top of the timer and use pre-scalar of 64
  */

  *portB &= 0xBF; // turn off bit 6

  *myTCNT1 = (unsigned int) (65536 - 350);
  *myTCCR1B |= 0x03; // turn timer ON with pre-scalar of 64 (0b011 => 0x03)
  while((*myTIFR1 & 0x01)!=1); // once overflow flag is 1, we will stop
  *myTCCR1B &= 0xF8; // turn timer to OFF (prof. Egbert says this is optional)
  *myTIFR1 |= 0x01;// clear overflow flag bit (by setting to 1 for some reason)
}
```

In the program on the prior page I simply created a function that turns on and off bit 6 of port B within the function. In the loop() i just call this genFreq() function. It turns the bit on for 30% of the 500Hz then off for 70% of the 500Hz. I hope I interpreted the question correctly. Below is proof that it compiled.

ANSI-C_Blink_timers_4

```
  *myTCCR1B = 0;
  *myTCCR1C = 0;
  *myTIMSK1 = 0; // Timer 1 should have no interrupts

  *portDDRB |= 0x80;  // Initialize GPIO PortB
}

void loop() {
  genFreq();
}

void genFreq() {
  /* TODO: generate frequency of 500 Hz, 30% up <---> 70% down */

  /*  period => 1/500 = 0.002 seconds = 2 milliseconds
   *  30% period for square wave => 0.3(1/500) = 0.6 milliseconds
   *  we want the delay to be 0.6 milliseconds for UP
   *  0.6 milliseconds = 600 microseconds
   *  600 microseconds/4 microseconds = 150
   *  we need to subtract 150 from the top of the timer and use pre-scalar of 64
   */

  *portB |= 0x40; // turn on bit 6 (0b01000000 => 0x40) // turn on bit 6

  *myTCCR1B &= 0xF8; // turn timer to OFF
  *myTCNT1 = (unsigned int) (65536 - 150);
  *myTCCR1B |= 0x03; // turn timer ON with pre-scalar of 64 (0b011 => 0x03)
  while((*myTIFR1 & 0x01)!=1); // once overflow flag is 1, we will stop
  *myTCCR1B &= 0xF8; // turn timer to OFF (prof. Egbert says this is optional)
  *myTIFR1 |= 0x01;// clear overflow flag bit (by setting to 1 for some reason)

  /*  period => 1/500 = 0.002 seconds = 2 milliseconds
   *  70% period for square wave => 0.7(1/500) = 1.4 milliseconds
   *  we want the delay to be  1.4 milliseconds for DOWN
   *  1.4 milliseconds = 1400 microseconds
   *  1400 microseconds/4 microseconds = 350
   *  we need to subtract 350 from the top of the timer and use pre-scalar of 64
   */

  *portB &= 0xBF; // turn off bit 6

  *myTCNT1 = (unsigned int) (65536 - 350);
  *myTCCR1B |= 0x03; // turn timer ON with pre-scalar of 64 (0b011 => 0x03)
  while((*myTIFR1 & 0x01)!=1); // once overflow flag is 1, we will stop
  *myTCCR1B &= 0xF8; // turn timer to OFF (prof. Egbert says this is optional)
  *myTIFR1 |= 0x01;// clear overflow flag bit (by setting to 1 for some reason)
}
```

Done compiling.

```
Sketch uses 926 bytes (0%) of program storage space. Maximum is 253,952 bytes.
Global variables use 25 bytes (0%) of dynamic memory, leaving 8,167 bytes for local variables. Maximum is 8,192 bytes.
```

5) Below is the code for this part, with comments in the shutterControl function explaining my math process

```
volatile unsigned char *myTCCR1A = (unsigned char *) 0x80; // Timer/Counter Control Register A
volatile unsigned char *myTCCR1B = (unsigned char *) 0x81; // Timer/Counter Control Register B
volatile unsigned char *myTCCR1C = (unsigned char *) 0x82; // Timer/Counter Control Register C
volatile unsigned char *myTIMSK1 = (unsigned char *) 0x6F; // Timer/Counter Interrupt Mask Register
volatile unsigned int  *myTCNT1  = (unsigned  int *) 0x84; // Timer/Counter Register (low & high)
volatile unsigned char *myTIFR1 =  (unsigned char *) 0x36; // Timer/Counter Interrupt Flag Register
volatile unsigned char *portDDRB = (unsigned char *) 0x24;
volatile unsigned char *portB =    (unsigned char *) 0x25;

void myDelay(unsigned long);

void setup() {
  /* Initialize Timer1 for NORMAL mode */
  *myTCCR1A = 0;
  *myTCCR1B = 0;
  *myTCCR1C = 0;
  *myTIMSK1 = 0; // Timer 1 should have no interrupts

  //*portDDRB |= 0x80; Question says this is already done
}

void loop() {

}

void shutterControl(int shutterSpeed) {
  // assuming shutter starts closed (off)

  *myTCCR1B &= 0xF8; // turn timer to OFF

  /*  base period => 1 second = 1000 milliseconds
   *  we want the delay to be 1000 milliseconds
   *  1000 milliseconds = 1,000,000 microseconds
   *  1,000,000 microseconds/64 microseconds = 15,625
   *  we need to subtract 15,625 from the top of the timer for 1 second
   *
   */
  long toSubtract = 15625;

  switch(shutterSpeed) {
  case 0:
    // no change
    break;
  case 1:
    toSubtract*(0.5); // 1/2
    break;
  case 2:
    toSubtract*(0.25); // 1/4
    break;
  case 3:
    toSubtract*(0.125); // 1/8
    break;
  case 4:
    toSubtract*(0.06666666667); // 1/15
    break;
  case 5:
    toSubtract*(0.03333333333); // 1/30
    break;
  case 6:
    toSubtract*(0.01666666667); // 1/60
    break;
  case 7:
    toSubtract*(0.008); // 1/125
    break;
  case 8:
    toSubtract*(0.004); // 1/250
    break;
```

```
  case 9:
    toSubtract*(0.002); // 1/500
    break;
  case 10:
    toSubtract*(0.001); // 1/1000
    break;
  default:
    //something
    toSubtract*(1); // nothing by default
  }

  *myTCNT1 = (unsigned int) (65536 - toSubtract);
  *myTCCR1B |= 0x03; // turn timer ON with pre-scalar of 64 (0b011 => 0x03)

  *portB |= 0x80; // turn shutter on (open)

  while((*myTIFR1 & 0x01)!=1); // once overflow flag is 1, we will stop
  *myTCCR1B &= 0xF8; // turn timer to OFF (prof. Egbert says this is optional)
  *myTIFR1 |= 0x01;// clear overflow flag bit (by setting to 1 for some reason)

  *portB &= 0x7F; // turn shutter off (closed)
}
```

And below is the program compiling: