SEED LAB Documentation: ASSIGNMENT 1

Malachi Contreras

Questions:

a. How many digital I/O pins do you have access to on the Uno? How many of these pins can provide pulse
width modulated (PWM) analog output?
There are 14 digital I/O pins, and 6 of them can provide PWM analog output.

b. Provide an examples where you might use the PWM functionality of those pins.

Motor speed control: adjusting the PWM duty cycle we can power the motor to rotate slower or faster based on the duty cycle.

c. How many analog I/O pins do you have access to on the Uno?
There are 6 analog I/O pins on the Uno.

d. What is the recommended supply voltage for the Arduino?
The recommended supply voltage is from 7-12v.

e. How many external interrupts are available on the Uno, and on which pins?
There are 2 external interrupts on pin2 and pin3.

f. What are the two functions that will always be a part of an Arduino sketch? Explain the purpose of each
of these functions.

The two functions that will always be apart of an Arduino sketch are *void setup()* and *void loop()*.

the *setup()* function runs only once when a sketch begins to initialize variables, pin modes and libraries (set up code) and the *loop()* function will then run continuously which allows the program to change and respond.(used to actively control board)

g. What is the purpose of the delay function in an Arduino sketch?
The purpose of the delay function is to make the board do nothing for a set amount of milliseconds.

h. What is the main drawback of using the delay function to control timing?

The main drawback of using the delay function to control timing is that the delay function will block the processor so that it cant do anything else such as read inputs or update other task.

i. What are the alternatives to using the delay function?
Some alternatives include: millis(), micros(), hardware timers(interrupts).

j. What is a pull-up resistor, why is it used, and how can it be implemented internally on the Arduino?
A pull-up resistor is a resistor that makes the pin read a stable HIGH so that when the switch closes it gives a reliable low signal and does not float or pick up noisy/half inputs.

Coding Section

Exercise 1a.

```
//Setup: LEDS attached to pins 10,11,12,13 into breadboard and pin 3 and 2 for
//gas and brake  push buttons respectively. The program makes the lights turn
//on and off in order at faster speed when the gas is pressed and slower speed
//when the brake is pressed.
```

```
// ==== LEDs ====
const int ledPins[] = {10, 11, 12, 13};
```

```
const int numLeds = 4;

// ==== Buttons ====
const uint8_t GAS_PIN  = 3;  // INT1
const uint8_t BRAKE_PIN = 2;  // INT0

// ==== Debounce time (us) ====
const unsigned long DEBOUNCE_US = 150000UL; // 150 ms

// ==== Interrupt flags ====
volatile bool gasPressedFlag  = false;
volatile bool brakePressedFlag = false;
volatile unsigned long lastGasUs  = 0;
volatile unsigned long lastBrakeUs = 0;

// ==== LED speed control ====
// Smaller = faster. Index 0 is slowest, last is fastest.
const unsigned long SPEED_TABLE[] = {1000, 800, 600, 450, 350, 275, 210, 160,
120, 90};
const uint8_t SPEED_MIN = 0;
const uint8_t SPEED_MAX = sizeof(SPEED_TABLE)/sizeof(SPEED_TABLE[0])  - 1;
uint8_t speedIdx = 4; // start near the middle (350 ms)

// ==== Animation state ====
// We implement the setup() intro animation as a repeating sequence of 8 steps:
uint8_t stepIdx = 0;            // 0..7, loops
unsigned long lastStepMs = 0;      // last time we advanced a step

// ---- ISRs ----
void onGasPress() {
  unsigned long t = micros();
  if (t - lastGasUs > DEBOUNCE_US) {
    gasPressedFlag = true;
    lastGasUs = t;
  }
}

void onBrakePress() {
```

```
  unsigned long t = micros();
  if (t - lastBrakeUs > DEBOUNCE_US) {
   brakePressedFlag = true;
   lastBrakeUs = t;
  }
}

void setup() {
  // LED pins
  for (int i = 0; i < numLeds; i++) {
   pinMode(ledPins[i], OUTPUT);
   digitalWrite(ledPins[i], LOW);
  }

  // Buttons & interrupts
  pinMode(GAS_PIN,  INPUT_PULLUP);  // pressed = LOW
  pinMode(BRAKE_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(GAS_PIN),  onGasPress,  FALLING);
  attachInterrupt(digitalPinToInterrupt(BRAKE_PIN), onBrakePress, FALLING);

  // We no longer do the blocking intro animation here.
  // The animation now runs continuously in loop() and its speed is adjustable.
}

void loop() {
  // --- Handle speed changes from button taps ---
  if (gasPressedFlag) {
   gasPressedFlag = false;
   if (speedIdx < SPEED_MAX) speedIdx++;  // go faster
  }
  if (brakePressedFlag) {
   brakePressedFlag = false;
   if (speedIdx > SPEED_MIN) speedIdx--;  // go slower
  }

  // --- Advance the animation based on current speed ---
  unsigned long now = millis();
  unsigned long interval = SPEED_TABLE[speedIdx];
```

```
  if (now - lastStepMs >= interval) {
   lastStepMs = now;

   // Compute which LED and whether to turn it on or off for this step
   if (stepIdx < numLeds) {
    // Steps 0..3: turn LEDs 0..3 ON
    uint8_t led = stepIdx; // 0..3
    digitalWrite(ledPins[led], HIGH);
   } else {
    // Steps 4..7: turn LEDs 3..0 OFF
    uint8_t led = (numLeds - 1) - (stepIdx - numLeds); // 3..0
    digitalWrite(ledPins[led], LOW);
   }

   // Next step (wrap 0..7)
   stepIdx++;
   if (stepIdx >= 2 * numLeds) {
    stepIdx = 0;
   }
  }
}
```

Exercise 2a.
(with interrupts)

```
//setup: attach encoder( clk ->pin2, Dt->pin7) Code provides a count when the
//encoder is moves adding when moved CW and subtracting when moved ccW

// ---- Pins ----
const byte APIN = 2;   // A / CLK (INT0)
const byte BPIN = 7;   // B / DT

// CW positive (set to -1 if reversed)
const int8_t DIR_POLARITY = +1;
```

```
// debounce (ms). Use 1–2 ms for real use.
const unsigned long ISR_DEBOUNCE_MS = 50;

volatile int  lastA, lastB;
volatile long position = 0;
volatile unsigned long lastIsrMs = 0;

void onAChange() {
  unsigned long now = millis();
  if (now - lastIsrMs < ISR_DEBOUNCE_MS) return;
  lastIsrMs = now;

  int thisA = digitalRead(APIN);
  int thisB = digitalRead(BPIN);

  // --- print format (A\tB on each change) ---
  if (thisA != lastA || thisB != lastB) {
    Serial.print(thisA); Serial.print('\t'); Serial.println(thisB);

    // --- Count logic turnery if () then +1 if true -1 if false) ---
    if (thisA != lastA) position += ((thisA == thisB) ? +1 : -1) * DIR_POLARITY;
    else                position += ((thisA != thisB) ? +1 : -1) * DIR_POLARITY;

    lastA = thisA; lastB = thisB;
  }
}

void setup() {
//setup serial monitor
  Serial.begin(9600);

//pin mode and interrupt
  pinMode(APIN, INPUT_PULLUP);
  pinMode(BPIN, INPUT_PULLUP);
  lastA = digitalRead(APIN);
  lastB = digitalRead(BPIN);

  attachInterrupt(digitalPinToInterrupt(APIN), onAChange, CHANGE);
```

```
  Serial.println("Interrupt demo: prints A\\tB on change, pos every 1s (ISR
debounced)");
}

void loop() {
  static unsigned long lastPrint = 0;
  if (millis() - lastPrint >= 1000) {
    noInterrupts(); long p = position; interrupts();
    Serial.print("pos = "); Serial.println(p);
    lastPrint = millis();
  }
}
```

Exercise 2b.
(without interrupts)

```
//Setup(same as in exercise 2a.)

// ---- Pins ----
const byte APIN = 2;   // A / CLK
const byte BPIN = 7;   // B / DT

// Flip to -1 if CW is negative with your wiring
const int8_t DIR_POLARITY = +1;

// l tiny debounce for encoders (us)
const unsigned long CHANGE_DEBOUNCE_US = 800;

int lastA, lastB;
long position = 0;
unsigned long lastChangeUs = 0;

void setup() {
  Serial.begin(9600);
  pinMode(APIN, INPUT_PULLUP);
  pinMode(BPIN, INPUT_PULLUP);
  lastA = digitalRead(APIN);
  lastB = digitalRead(BPIN);
```

```
  Serial.println("Polling demo: prints A\\tB on change, pos every 1s");
}

void loop() {
  int thisA = digitalRead(APIN);
  int thisB = digitalRead(BPIN);

  if (thisA != lastA || thisB != lastB) {
    unsigned long t = micros();
    if (t - lastChangeUs >= CHANGE_DEBOUNCE_US) {
      lastChangeUs = t;

      // --- Your print format ---
      Serial.print(thisA); Serial.print('\t'); Serial.println(thisB);

      // --- Count logic tunery
      if (thisA != lastA) position += (thisA == thisB) ? +1 : -1;
      else                position += (thisA != thisB) ? +1 : -1;

      lastA = thisA; lastB = thisB;
    }
  }

  // Timed pos print
  static unsigned long lastPrint = 0;
  if (millis() - lastPrint >= 1000) {
    Serial.print("pos = "); Serial.println(position);
    lastPrint = millis();
  }
}
```