Mechatronics

Badger Invitational
Hamilton Middle School
2019-02-23

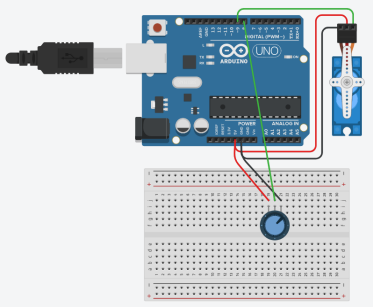
Using your browser, visit the website `http://mechatronics.blaise.zone`. Here you will find links to all of the Tinkercad circuits referred to in this exam. After the competition, the correct answers will be released on this same website.

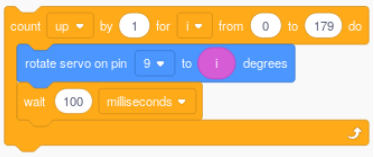
1 Written Test

You will have 25 minutes to complete this portion of the exam. There are four questions, each worth 25 points for a total possible score of 100 points.

1.1 servo

Using Tinkercad, open the "servo" circuit. You may choose to use text or blocks to program the Arduino.






```
#include <Servo.h>

Servo servo;

void setup() {
  servo.attach(9);
}

void loop() {
  for (int i = 0; i <= 180; i++) {
    servo.write(i);  // degrees, 0 to 180
    delay(100);  // ms
  }
}
```

1.1.1 10 points

Your Arduino has been pre-programmed to remind you how to use the Servo library. As written, the code will simply move the motor from 0 to 180 degrees and then start over. Roughly how long will the motor take to accomplish one rotation? Modify the code so that the pattern repeats at 0.5 Hz.

As written, the motor will take 18 seconds to accomplish one rotation. To repeat the pattern at 0.5 Hz, change the delay from 100 ms to (5000/180)=28 ms. For a super-accurate repeat frequency, use millis() instead.

1.1.2 5 points

The middle pin of the potentiometer is wired to pin 8 of the Arduino. Is this appropriate? If not, choose an appropriate pin and explain your reasoning.

Pin 8 is not appropriate. As the potentiometer is rotated, the middle pin will sweep from zero to five volts. An analog input pin is required to measure the analog output of the potentiometer. Pin 8 cannot be used as an analog input. Pins A0 though A5 are all appropriate choices.

1.1.3 10 points

Modify your code such that the potentiometer can be used to control the position of the servo motor. Ensure that the dynamic range of the potentiometer (0 to 5 V) agrees with the dynamic range of the servo motor (0 to 180 degrees). Recall that the Arduino has a 10 bit analog-to-digital converter.

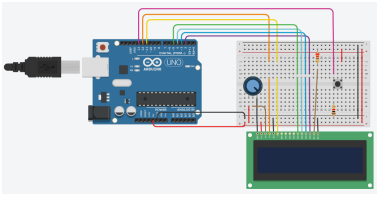
With the reminder that the Arduino Uno has a 10 bit analog-to-digital converter, we can calculate that the analogRead() function will return an integer between 0 and ((2^10)-1)=1023. An input of zero volts will give us integer 0, and an input of 5 volts will give integer 1023. The conversion factor, therefore, is (180/1024)=0.176.

Please visit `http://mechatronics.blaise.zone/2019-02-23/hamilton.html#servo` for links to the modified circuits.

For students using Tinkercad blocks, the map function can also be used to answer this question.

1.2 stopwatch

Your friend has designed a simple stopwatch using an Arduino, button, and LCD display. How clever of her! The circuit works perfectly, but the code is behaving strangely. She needs your help to debug the project. Open the stopwatch circuit in Tinkercad. You may choose to use text or blocks to program the Arduino.




```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int button = 13;
boolean running = false;
unsigned long startTime;

void setup() {
  lcd.begin(16, 2);
  pinMode(button, INPUT);
}

void loop() {
  if (digitalRead(button) == HIGH) {
    running = true;
    startTime = millis();
    // wait for button to be released
    while (digitalRead(button) == HIGH) {
      delay(100);
    }
  }
  delay(10);
  if (running) {
    lcd.setCursor(0, 0);
    lcd.print((millis() - startTime) / 1000.);
  }
}
```

1.2.1 10 points

The stopwatch starts correctly when the button is pressed the first time. Subsequent button presses reset the stopwatch. This is not intended behavior! Your friend wants her stopwatch to *pause* when the button is pressed a second time.

Briefly, explain to her *why* your friends' code doesn't work as intended. Be sure to describe the relationship between button presses and the "running" variable.

Almost everything is correct in the given circuit. The only problem is that, after the first button press, the "running" variable is never set back to false (zero). As written, running is set to true (one) every time the button is pressed. Since running never goes false (zero), the number keeps being updated no-matter-what. Your friend should toggle the running variable, rather than just setting it to true (one) inside the if statement.

1.2.2 10 points

Fix your friends' code. Describe what you changed, and why that change fixes the problem.

Please visit `http://mechatronics.blaise.zone/2019-02-23/hamilton.html#stopwatch` for links to the modified circuits.

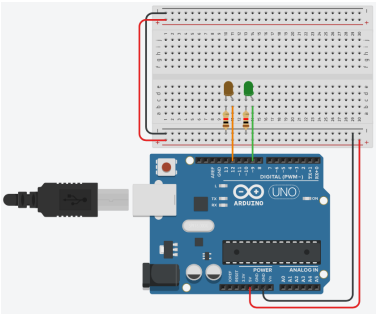
1.2.3 5 points

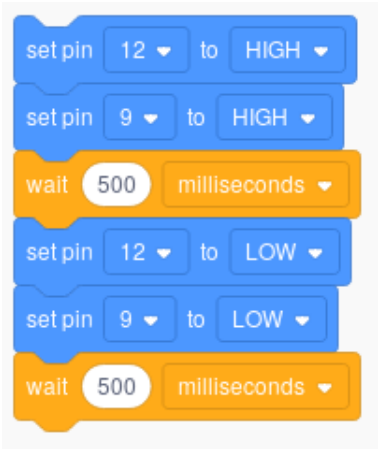
Why did your friend add a while loop?

The inner while loop ensures that the button is released before letting the Arduino continue. Without the while loop, the code would continue and eventually loop around to the (if button pressed) statement multiple times. This might lead to unexpected behavior where a single button press event results in the running variable being updated several times.

1.3 blink 2

Using Tinkercad, open the "blink2" circuit. You may choose to use text or blocks to program the Arduino.






```
int led1 = 12;
int led2 = 9;

void setup()
{
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
}

void loop()
{
  digitalWrite(led1, HIGH);
  digitalWrite(led2, HIGH);
  delay(500);
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  delay(500);
}
```

1.3.1 5 points

Blink both LEDs, such that the green LED is on when the orange LED is off and vice versa.

Simply change the HIGH to LOW and vice versa for one of the output pins.

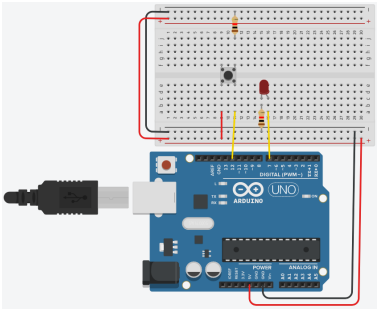
1.3.2 20 points

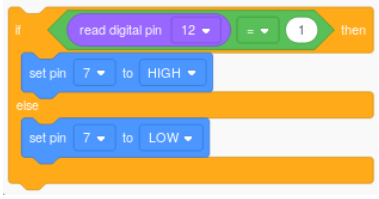
Change the frequency of the green LED such that it is blinking three times slower than the orange LED. Keep the duty cycle of both LEDs at 50 %.

Please visit `http://mechatronics.blaise.zone/2019-02-23/hamilton.html#blink2` for links to the modified circuits.

1.4 switch

Using Tinkercad, open the "switch" circuit. You may choose to use text or blocks to program the Arduino.






```
int led = 7;
int button = 12;

void setup()
{
  pinMode(led, OUTPUT);
  pinMode(button, INPUT);
}

void loop()
{
  digitalWrite(led, digitalRead(button));
}
```

1.4.1 10 points

Consider the following definitions:

A *toggle switch* changes the state every time it is depressed. Once released, the new state (on or off) is maintained until the switch is pressed again. Example: common light-switch.

A *momentary switch* is on only while being pressed. Once released, the switch automatically turns off again. Example: computer keyboard key.

Is the button acting as a momentary switch or a toggle switch? Is this relationship defined by the circuit, or by the Arduino code? Explain how you know.

The button acts as a momentary switch. The relationship is defined by both the code and the physical construction of the button. Any answer that demonstrates clear understanding of that relationship is acceptable.

1.4.2 10 points

Without changing the circuit, modify your Arduino code so that the button acts like a toggle switch. Describe your approach.

Please visit `http://mechatronics.blaise.zone/2019-02-23/hamilton.html#switch` for links to the modified circuits.

1.4.3 5 points

How could you change the brightness of the LED?

Change the brightness by changing the resistor. More resistance for a dimmer LED. Less resistance for brighter LED. You could also conceivably use pulse width modulation.

2 Whatchamacallits

You will have 25 minutes to complete this portion of the exam. Complete as many of the following exercises as you can. There are 100 points possible.

Some of these questions use Tinkercad, and others require the usage of real hardware. If you prefer to program using blocks, you must export your Tinkercad code and upload it using the Arduino IDE. If you have never worked with real Arduinos, we will help you get started.

2.1 traffic light - 10 points

Using the provided components, made an LED traffic light. Place three LEDs: green, yellow and red. Make them blink in a specific pattern: 1) long green, 2) short yellow, 3) long red, then repeat.

Visit http://mechatronics.blaise.zone/2019-02-23/hamilton.html#traffic-light for solution.

2.2 rainbow - 10 points

Using Tinkercad, make an RGB LED cycle through the rainbow (Red, Orange, Yellow, Green, Blue, Indigo, Violet) and back again.

Here is a table of values for each color:

	red	green	blue
Red	255	0	0
Orange	255	127	0
Yellow	255	255	0
Green	0	255	0
Blue	0	0	255
Indigo	75	0	130
Violet	148	0	211

Visit http://mechatronics.blaise.zone/2019-02-23/hamilton.html#rainbow for solution.

2.3 dancing motor - 10 points

Turn on a motor for one second, then off for one second, then on for two seconds, then off for two seconds, then on for three seconds, then off for three seconds, and so on repeating the process to 100 seconds.

Visit http://mechatronics.blaise.zone/2019-02-23/hamilton.html#dancing-motor for solution.

2.4 click counter - 20 points

Using Tinkercad, make a click counter. Using an LCD screen as a display, create a circuit that shows how many times a button is depressed. The circuit should count each click separately, even when the button is held down for a prolonged period.

Visit http://mechatronics.blaise.zone/2019-02-23/hamilton.html#click-counter for solution.

2.5 binary counter - 20 points

Using the provided components, make a three-bit binary counter. Use three LEDs. Count from zero to seven, then repeat. Spend roughly one second on each number.

Visit http://mechatronics.blaise.zone/2019-02-23/hamilton.html#binary-counter for solution.

2.6 7-segment - 30 points

You will be given an Arduino with a pre-wired seven segment display. Each segment is wired to a separate pin (see documentation). Program the Arduino to count from zero to nine, spending around one second on each number.

Visit http://mechatronics.blaise.zone/2019-02-23/hamilton.html#7-segment for solution.