TSK x REVO

3.0:
Week 4! This week we will be all about how to use interrupts, how to control big things with your little (micro!) controller, and how you could put it all together to build some controls.

3.1:
Interrupts are a key tool for being able to control things, accept input, or respond to the world. Interrupts let you use less core time by not having to constantly poll buttons. Lets look at this example:

```
int time;

void setup(){
pinMode(10, INPUT);
digitalWrite(10,HIGH);//turn on pull up resistor
pinMode(11, OUTPUT);
}

void loop(){
digitalWrite(11, HIGH);
delay(time);
digitalWrite(11, LOW);
delay(time);
if (digitalRead(10)==LOW){
time--;
}
}
```

Now imagine there is an LED attached to pin 11, and a button on pin 10. If you want to actually detect the button press there is a narrow window of opportunity to press the button. It is a few cycles wide (in time) after the LED goes low. You could also hold the button down, but it would be 2 seconds before the system did anything! Imagine if this was an alarm clock and you had to wait two seconds before you could adjust the time your alarm went off!

A much better way to handle this is with interrupts. Interrupts are exactly what they sound like- they interrupt the program to do something when they are called. Interrupts can be disabled with “noInterrupts();”, and enabled with “Interrupts();”. By default, they are on. Most arduinos have two inturrupt pins, and each pin can handle a separate interrupt based on one of several conditions.

1. LOW – any time the pin is low
2. RISING – any time the pin is pulled from low to high
3. FALLING – any time the pin is pulled from high to low
4. CHANGE – any time the pin changes from high to low or vice versa

Unfortunately, these are digital pins 2 and 3, which are used by vUSB! We can use these pins as long as we take off the VUSB hardware, but that might be a pain, so we won't use them for now. Here is an example though:

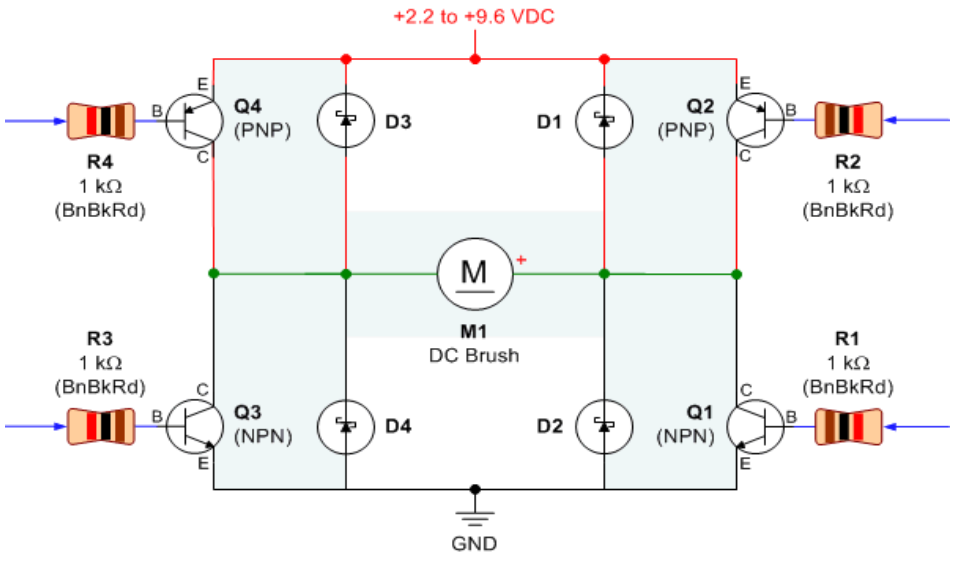
```
volatile int time;

void setup(){
attachInterrupt(0, button, FALLING);//interrupt on pin #2
digitalWrite(2,HIGH);//turn on pull up resistor on pin 2
pinMode(11, OUTPUT);
}

void loop(){
digitalWrite(11, HIGH);
delay(time);
digitalWrite(11, LOW);
delay(time);
}

void button(){
time--;
delay(100);
}
```

This will allow the button to be pressed whenever and have the same effect! Note that I added a delay in the interrupt function. This is because if we had this interrupt without this delay, it would run many times during one button press! Since this is not the desired behavior here, I added the delay so that there would be a .1s delay between button detections.

3.2:
Now, controlling big things with little things. There are several ways to do this, and the ones I will describe here are transistors, relays, and a collection of transistors called an H-bridge.

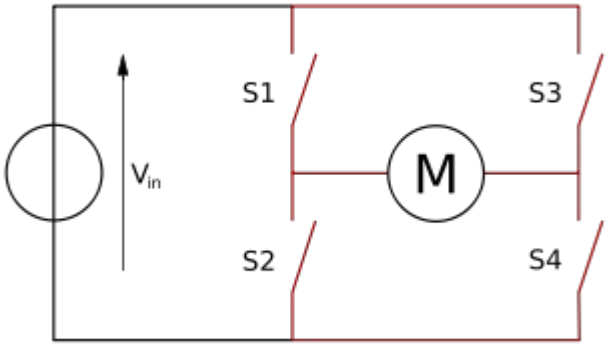
Transistors are devices that can handle a lot of current through them. They are controlled by small currents applied to (what is called in bipolar transistors) the base. In some transistors, more current is more current, and in others less current is more current. We will look at these a bit more in H-bridges!

H bridge:



Here we see we have two kinds of bipolar junction transistors (BJT)s, NPN and PNP. We need two kinds here because of the “side” that they are switching. PNP transistors turn on when the base (B) is pulled low relative to the emitter (E). We can see that these are the ones on top, which makes sense if we assume no negative voltages in our circuit, because if the emitter was at ground, there would be no way to turn them on. PNP actually refers to the doping of the semiconductor layers in the transistor, but another way to remember its function via its circuit diagram is that it is “proudly in pointing”, while a NPN is “not pointing in”. NPN transistors turn on when the base voltage is pulled low compared to the emitter, so it makes sense to have them where we have them in this diagram.

Now, lets imagine a more generalized H bridge with switches:



This is equivalent to the circuit above, but it has switches instead of transistors. What would happen if we closed switch 1 and 3? 2 and 4? 1 and 4? 3 and 2? lets make a table:

First switch	Second switch	Result
1	2	Short circuit
1	3	Nothing happens
2	4	Nothing happens
3	4	Short circuit
1	4	Clockwise motor rotation
3	2	Counterclockwise motor rotation

An H bridge works the same way, but with transistors. Here is a table for the “useful” outputs:

Q1	Q2	Q3	Q4	Result
Off/ground	Gnd	power	Off/power	CW rotation
Power	Off/power	Off/gnd	gnd	CCW rotation

Note: references to CW/CCW rotation are to show that the motor will go in opposite directions. Rotation of actual motor will depend on your frame of reference.

3.3:
Relays are also handy for controlling large voltages. “Relay” is an appropriate name for the device

because it literally relays information from one circuit to the other, while keeping the two electrically isolated. Mechanical relays use a switch and an electromagnet. The electromagnet actuates the switch, an is activated by a low voltage low current signal-something the microcontroller can supply if you choose the right relay. There are now solid state relays that do not have moving parts, which makes them more reliable. These work by magic.

Just kidding. Basically there are some FETS (field effect transistors, also magic) and they are turned on when a light sensor receives light from an led. The LED and sensor are packaged inside of the relay. Here the microcontroller turns on the LED, and that switches on the larger load. The two circuits are said to be optocoupled, or optically coupled, but they are electrically independent.