Lab 3: Motion

For the rest of this semester you will be designing SIXT33N, a mischievous little robot who *might* just do what you want - if you design it correctly. Our end goal is to have S1XT33N listen to your voice commands and execute the corresponding drive command. Over the course of this semester, we will be implementing different parts of this project, from the circuits to the algorithms necessary for voice classification and control. In Lab 2, you learned about digital/analog interfaces, which is how your Launchpad, which will serve as the brain of S1XT33N, collects information from and interacts with the world. In this lab, **you will build the legs** of SIXT33N: you will be designing SIXT33N's wheels. Later you will learn how to control the robot with voice commands using a microphone emulator in TinkerCAD. The wheels will be driven by two DC motors which you are going to build in this lab. The goal of this lab is to **construct motor controller circuits**.

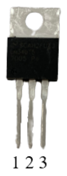
Part 1: Regulator Circuits

SIXT33N, like any other electronics, requires power to run. If we were only using it at the lab stations, we could just hook it up to the DC power supply. However, because we want to let SIXT33N roam freely on the ground without confinement to a lab station, we need to power it with a more portable source: batteries. We only have access to 9V batteries, however, we need a way to generate the 5V we need for Arduino logics from the 9V battery. This is where the voltage regulators come in.

5V Regulator Circuit

The LM78xx series are also voltage regulators with several fixed output voltages. The LM7805 is really just meant for 5V regulation (hence the 5 in LM7805), so we use it to regulate the 9V input down to a 5V output. Adding "decoupling" capacitors to the input and output is useful for removing all sorts of noise and interference from the 9V supply and the 5V output to make them both more stable.

The pinout is reproduced below for your convenience. The image is taken from a front angle where the metal tab that sticks out is at the back of the regulator. To generate stable 5V output, we will connect the input and output to decoupling capacitors shown in Figure 1, which can be found in Figure 9 in the datasheet.



INPUT	1
GND	2
OUTPUT	3

Table 1: 5V Regulator

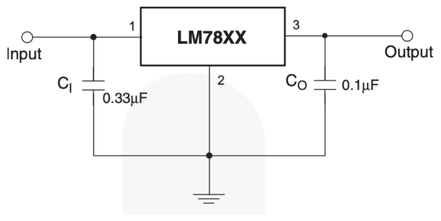


Figure 1: Fixed-Output Regulator

Now you are ready to do the first part of the lab! Go to the Jupyter notebook and complete part 1.

Part 2: Motor Drivers

S1XT33N uses 9-Volt DC motors. This means the maximum voltage that can be delivered to the motors is 9V, but the motors will move (albeit more slowly) with voltages less than 9V. There is some minimum voltage required to deliver enough power to the motors to overcome the static friction and start them, but after that point, we treat the motor speed as approximately linear with the applied voltage (this will be the basis of the system model you develop in the next lab).

Because it is difficult to use the Arduino to control a true DC signal within 0V and 5V (e.g. 3.2V), we will instead make use of its PWM function. A PWM, or pulse-width modulated, signal is a square wave with a variable duty cycle (the proportion of a cycle period for which the power source is turned on, or logic HIGH) shown in Figure 2. PWM is used to digitally change the average voltage delivered to a load by varying the duty cycle. If the frequency is large enough, the on-off switching is imperceptible, but the average voltage (V_{AVG}) delivered to the load changes proportionally with the duty cycle. Hence, changing the duty cycle corresponds to changing the DC voltage supplied to the motor. For example, you can get average voltage equals to 3.2V by setting the duty cycle to $\frac{3.2}{5} = 64\%$.

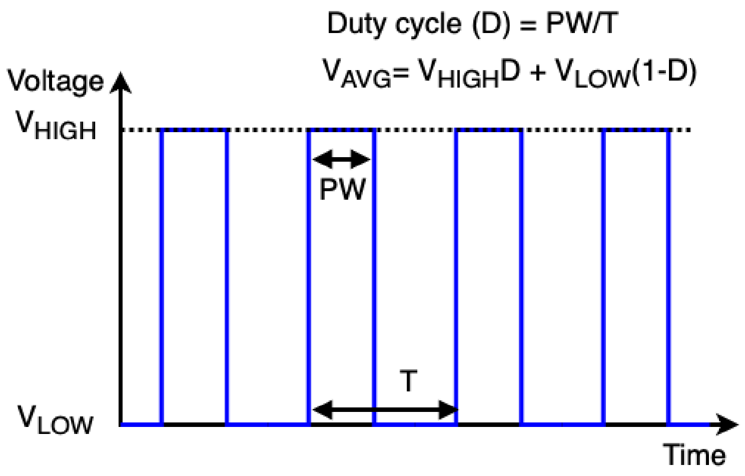


Figure 2: PWM Signal

We will use the Arduino's digital (PWM) pins to set the duty cycle for each motor's PWM signal. However, the Arduino's logical HIGH[1] voltage is 5V, which is not enough to directly drive the motors: even if your motor does turn on at 5V when you test it with the power supply, the Arduino cannot supply enough current to power the motors. This is where the motor driver circuits come in.

[1]The logic-level signals are denoted as HIGH and LOW, where HIGH=5V and LOW=0V for the Arduino.

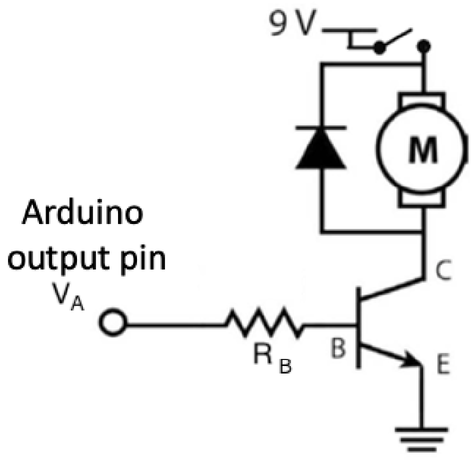
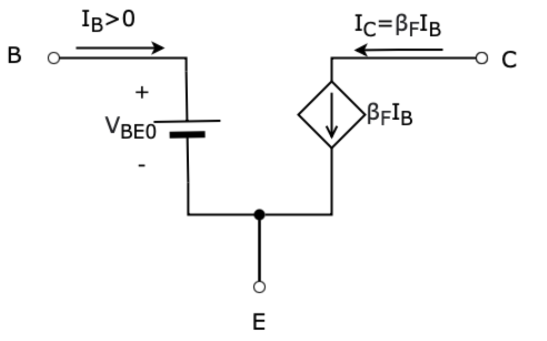
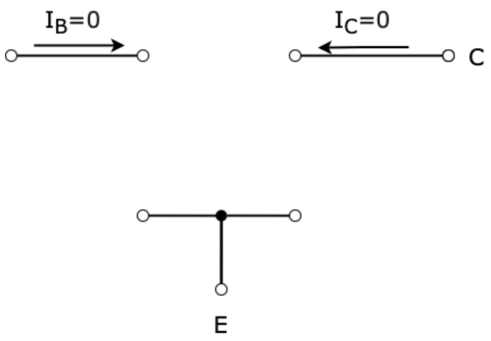


Figure 3: Motor Controller Circuits



(a) Model of BJT in ON mode (when Arduino output pin is HIGH)



(b) Model of BJT in OFF mode (when Arduino output pin is LOW)

Figure 4: Model of NPN BJT in Different Modes

The PWM pin from the Arduino is connected via a resistor to the “Base(B)” of an NPN bipolar junction transistor (BJT). This transistor, in reality, behaves a bit differently from the NMOS with which you are familiar, but for this class, you may assume that it is functionally the same as an NMOS, behaving as a voltage-controlled switch. On the BJT, the three terminals are analogous to those of an NMOS: the “Base (B)” is the gate, the “Collector (C)” is the drain, and the “Emitter (E)” is the source.

BJT in Figure 3 is switching between **ON** and **OFF** modes when the output voltage from Arduino V_A is HIGH (5V) and LOW (0V) respectively. The model for both ON and OFF states are shown in Figure 4. When the BJT turns on, V_{BE} can be modeled as a fixed voltage source with voltage value V_{BE0} which is usually between 0.6V and 0.8V. For NPN BJT in TinkerCad, the value of V_{BE0} is approximately 0.8V in ON mode. In ON mode, there is a **Current Control Current Source** modeled between“ Collector(C)” and “Emitter(E)”, i.e., current at the “Collector (C)” is an amplified version of current at the “Base (B)” (notice that I_B has to flow into the “Base(B)” for the relation $I_C = \beta_F I_B$ to hold.) β_F is called the **Common-Emitter Current Gain**, e.g. β_F is around 100 to 200 in our circuits. When the Arduino output pin V_A is HIGH (5V), the voltage across the resistor R_B in Figure 3 is $V_{HIGH} - V_{BE0}$. So we can calculate the current flowing into the “Base (B)”, I_B, and the current flowing into the “Collector (C)”, I_C, as

$$I_B = \frac{V_{HIGH} - V_{BE0}}{R_B} \tag{1}$$

$$I_C = \beta_F I_B = \beta_F \frac{V_{HIGH} - V_{BE0}}{R_B} \tag{2}$$

From Equation (1) and (2), we know that the resistor in the motor controller circuit (Figure 3) is used for defining the current flowing through the Arduino output pin, BJT, and motor. Therefore, we should carefully choose the value of the resistor considering the maximum current that BJT and Arduino pin can handle and the current required to drive the motor. As you will see in the lab, we can also control the rotation speed of the motor by alternating the resistor value. As we increase the value of the resistor, the current flow through the motor decreases which slows down the motor rotation.

Ignoring the diode in the schematic, the low-power, low-voltage signal from the Arduino switches the BJT on and off: when the Arduino signal is HIGH (5V), the BJT is on, we have 9V-V_{CE} dropped across the motor (V_{CE} is the voltage across the current control current source in Figure 4a) and current can flow through the motor and BJT. When the Arduino signal is LOW (0V), the BJT is off. As shown in Figure 4b, the BJT is open at all terminal. Therefore, no current is allowed to flow into the motor and there is no voltage drop across the motor (0V). Our Arduino PWM wave that only spanned 0V-5V thus has been amplified to a signal that roughly spans 0V-9V (neglecting the V_{CE}) across the motor!

Moreover, the BJT is capable of handling far more current than the Arduino pins can tolerate, allowing the necessary hundreds of mA to flow through the motor so it can run properly. The Arduino only needs to supply the small amount of current the BJT needs flowing into its base terminal to function (this is a key difference between the NPN BJT and NMOS) and the BJT will amplify that current to be used by the motor. With this circuit, we have thus supplied the motor with a high-power PWM signal.

You may be wondering why we have the diode connected across the motor. We add this diode because motors have inductive properties; the current flowing through the motor cannot change instantaneously. When we turn off the BJT, we are cutting off all current from flowing through the BJT. Above, where we ignored the diode, we assumed the motor could instantly go from an ON to OFF state. However, because the motor has inductive properties, if we did not have the diode in the circuit, the current through the motor would have to instantaneously drop from hundreds of mA (ON) to 0A (OFF), causing a huge voltage spike at the Collector of the BJT, which would be disastrous. In order to prevent this from happening, we add the diode as a pathway to allow current to flow through it when the BJT is turned off, enabling the current to drop over time instead of instantaneously. Once the current decreases sufficiently, the diode turns off, and the motor is just connected to an open circuit, meaning no current will flow across the motor. This results in no voltage drop across the motor.

We have simplified our analysis and description of the different components we use in the motor circuit drastically so as to remain in scope for this class. If you would like to learn more about these components after doing the lab, please read the extra reading note for this lab for more information.

You are now ready to start Part 2! Go to the Jupyter Notebook and complete the rest of the lab.

References

Original Project Part 1 notebook written by Nathaniel Mailoa and Emily Naviasky (2016).

Horowitz, P. and Hill, W. (2015). *The Art of Electronics*. 3rd ed. Cambridge: Cambridge University Press, ch 4.

Horowitz, P. and Hill, W. (2015). *The Art of Electronics*. 3rd ed. Cambridge: Cambridge University Press, ch 2.

Sedra, A. S. and Smith, K. C. (2015). *Microelectronic Devices and Circuits*. 7th ed. Oxford: Oxford University Press, ch 3.

Sedra, A. S. and Smith, K. C. (2015). *Microelectronic Devices and Circuits*. 7th ed. Oxford: Oxford University Press, ch 6.

Notes written by Mia Mirkovic (2019)
Notes edited by Hossein Najafi, Steven Lu, Yi-Hsuan Shih (2021)