Module: Arduino Input/Output (I/O) Pins

Module Outline

In this module you will be learning how to use the Digital I/O pins on the Arduino effectively. Digital? What does digital mean in this context? So far voltages and current are physical processes that you have learned to model – what does it mean to say that a signal is digital? The Arduino board has analog input pins that you will learn to use in a subsequent module – how is digital different from analog?

The history of the Digital Age is a fascinating one and the personnel in the ECE department played no small role in this history (and TI too). The achievement of being able to miniaturize devices that are to first order electronically controlled switches truly changed how humans live their lives. This ability allowed devices based on the concept of ON and OFF to be put together to build computers that are now an integral part of our lives. Using the digital I/O pins will help you see how this abstract Boolean 2-state notion of ON or OFF maps onto real voltage signals.

After constructing the chassis of your car and mounting the motors, the only way to get the vehicle to move is to physically connect the motor terminals to either the power supply or battery by hand or with hand-operated switches. The autonomous in autonomous vehicle means that your car must navigate on its own responding only to the input of sensors and other devices that monitor some parameter of the environment.

Multi-Valued Logic – our current computers are based on a 2-valued logic – a concept dating back to at least the ancient Greeks. Everything is either TRUE or FALSE, ON or OFF, but there are people looking into the value of a construct that has more than just 2 states. I wonder what advantage there would be to having a 3-state logic system with TRUE, FALSE, and MAYBE. Or 4 or 5 or… isn’t a signal we call an analog signal just an system of an infinite number of states?
At the hardware level a digital signal is just a voltage that either has a “high” value or a “low” value. These values are different for different families of devices – the terminology family refers to the fabrication process that made the digital hardware. The Arduino board and clones are made to interface with two logic types or families – TTL transistor-transistor-logic and CMOS complementary-metal-oxide-semiconductor. Each has a different definition for what constitutes a HIGH and LOW voltage. The SparkFun website https://learn.sparkfun.com/tutorials/logic-levels has a marvelous and more detailed description.

Any of the logic families works with a range of voltages – the TTL family used a voltage range from 0-5V and the CMOS most typically uses a range from 0-3.3V. This is where things get a little confusing because the hardware dealing with the digital signals must define ranges that are considered “high” and ranges that are “low”. In most technologies there is a middle range of “I don’t know” where the hardware behaves unpredictably. If a digital device is connected to another digital device this is not a problem since the output levels are always in the appropriate ranges. It is more problematic when interfacing to analog hardware.

The digital I/O pins on the Arduino are actually quite flexible. They were constructed so that the input pins interface with as many different types of devices as possible. On output the Arduino consistently provides a HIGH voltage very close to 5V and a LOW close to 0V. The output pin interface on the board are well-designed so that the voltages remain the same for a wide range of loads acting like an ideal voltage source that is either ON or OFF as the pin is connected to hardware that draws 40mA or less.

The digital inputs can be considered perfect measuring devices that draw no current and interpret the voltage range 2.6 – 5V as a HIGH and 0 – 2.1 V as a LOW.
Output

Most of your designs will involve using the digital I/O pins in output mode so the questions are primarily directed at using them correctly. For the robot cars you will need to turn the motors on and off in response to other inputs to steer the vehicle. As you will see by turning these digital output pins on and off really quickly you can also control the speed of the motors.

Instead of beginning with controlling your motors let’s begin by turning ON and OFF an LED. The motors will behave in a similar fashion but are more difficult to hook up. So to get use to the commands needed to control the digital I/O pins you will use an LED for now. Using the online version of the SIK guide or the Sparkfun experiment webpage [https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32/introduction-sik-arduino-uno](https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32/introduction-sik-arduino-uno) setup the circuit described in Experiment 1: Blink an LED. Below is a physical layout of the experiment for your convenience.

**Fritzing?** What a great word. Fritzing is an organization dedicated to making learning electronics interesting and fun. One of the apps lets you draw cool physical diagrams. This one was created by Sparkfun. Check out their website for cool ideas and tutorials. [http://Fritzing.org](http://Fritzing.org)
Question 1: From the physical diagram draw a schematic of the circuit. Rather than draw the Arduino you may simply label the signals connected to the Arduino – one connection is made to the GND pin on the board and the other is made to the digital I/O Pin 13. It is given in the SIK manual so you CAN just copy it over or you can challenge yourself and try to draw without looking. Either way designate on the schematic the points labeled A, B, C, D, and E.

To make the LED turn on you need to apply enough voltage across the terminals. To turn the LED on and off the voltage across its terminals much go high and low. This is where the digital I/O pins on the Arduino become useful. A program running on the Arduino board that raises and lowers the voltage one of the I/O pins is simple to write.

- Run the Arduino program on the computer that will be used to download the program and power the board.
- To find the program navigate the menu items File > Examples > SIK_Guide_Code > Circuit_01 and a window will open up with the software needed to blink the LED by using the Digital Pin 13 as an output.
- Attach the cable in your kit that has a mini USB connector on one end. The mini USB is connected to the board, the other end goes into any available USB port on the computer.
Question 2: Look at the Arduino Code and all of the comments that explain how to build the circuit, connect it to the processor board, run the 'sketch', as well as a short explanation of the structure of the code. Please read them – below is a condensed version without the comments so you can see more clearly the code structure. Using the information in the comments or on the Arduino website explain the function of the setup and loop portions of the code.

    /*
    SparkFun Inventor's Kit
    Example sketch 01 - just the facts
    */

    void setup()
    {
      pinMode(13, OUTPUT);
    }

    void loop()
    {
      digitalWrite(13, HIGH); // Turn on the LED
      delay(1000);            // Wait for one second
      digitalWrite(13, LOW);  // Turn off the LED
      delay(1000);            // Wait for one second
    }

Question 3: Either from reading the comments or by looking in the SIK Guide describe how the statement pinMode is used and what the parameters 13 and OUTPUT specify.

Question 4: Again by reading the comments or the SIK Guide explain how the functions digitalWrite and delay are used and what their parameters specify.
**Question 5:** Before actually uploading the program predict the behavior of the LED.

- Upload the program to the board by clicking the upload button. Mouse over the different buttons along the top panel – the one with the arrow is the upload button. **NOTE:** Common reasons for this step to fail are the Arduino does not know which of the many boards you are using or the COM port (USB port you plugged the board into) is incorrect. Both problems can be addressed by looking in the Tools option in the menu bar.

**Question 6:** If you uploaded the program properly to the microcontroller board the circuit should have started working. Describe the behavior of the LED noticing its brightness and the length of time it is in the ON and OFF states.
Question 7: Through the parameter passed to the delay function you have control over the time that the LED is ON and OFF. Lower the values, but for now keep them the same. Experiment with values between 1-1000. Find the values where your eye can no longer see the LED turn on and off. Describe what you see for all of the values you tried.

<table>
<thead>
<tr>
<th>Delay value</th>
<th>Blinking</th>
<th>Brightness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Question 8: Set the value in the delay functions both to 50. The LED should not be blinking but should be steadily on to your eyes. Vary the value of the value passed to the delay following the digitalWrite function that sets output on Pin 13 HIGH from 50 to 1. Describe what happens.

Question 9: Reset the value in the delay functions both to 50. Now vary the value of the value passed to the delay following the digitalWrite function that sets output on Pin 13 LOW from 50 to 1. Describe what happens.
Question 10: Make an educated guess about how the voltage varies with time using a sketch for 3 different pairs of delays. Put as much detail as you can. Voltage levels, time dependence. Make your best guess as we will be checking it in the next section of these procedures with the oscilloscope.

From the experimenting you have done so far it should be clear that the voltage signal that is driving the LED varies with time. It is always a good habit to check that the signals you generate.

✓ Connect the oscilloscope to measure the voltage across the LED.

Question 11: What is the oscilloscope showing?
**Question 12:** Vary the parameters passed to the `delay` function in different combinations and sketch the resulting signals. Try making the values different and make them very small or large. Notate on each sketch the behavior of the LED.

There is a shortcut to creating square wave pulses without resorting to manually setting the time the output signal is high and low if all that you want is a square wave with a FIXED period but a variable duty cycle which is to use a special function in your program.

- Make a new Sketch by saving the Blink Sketch to a file – name of your choice. Now all of the edits will be to this new file.
- Delete all of the statements in the loop portion of the program and replace them with the statement – `analogWrite(13,128);`
- Upload the new program to the board. The LED should not be blinking because the `analogWrite` does not work with all of the digital I/O pins.
- Look on the board at the pins labeled DIGITAL (PWM~). The pins with the ~ next to their number are the only ones that can be used with the `analogWrite` function. Choose any of them.
- Change all of the statements that reference the pin number and change the 13 to the number you chose in the previous step.
- Rewire your LED circuit to use the new signal.
- Monitor the voltage across the LED with the oscilloscope.

**Question 13:** On a new graph draw the waveform – notate the amplitude, period, and duty cycle.
Question 14: Now vary the second number from 128 to 64. What changed about the resulting signal and what did not.

Question 15: Without experimenting guess the range of values for the second number. Try both setting it to 0 (which is the minimum) to the maximum value you just guess to see if you are correct.
Conclusions

During this experiment you should have noticed that the LED visibly turned ON and OFF if the delay values were large enough. If the delay values were small enough your eyes could no longer distinguish the changing state of the LED and it averaged over the amount of time the LED was on. Even if the LED was ON for 50 ms and OFF for 50 ms (period of 100ms) the LED looked BRIGHTER than if the LED was ON for 10 ms and OFF for 40 ms (period of 50ms).

**Question 16**: How would the LED brightness compare if both parameters passed to the `delay` functions were 50 or they were both 25? Same? Different? Explain.

**Question 17**: When using the `analogWrite` function the period of the

What if the motors on your car were connected in the appropriate manner to the same digital outputs. The physics of the process is different. The brightness of the LED changes in response to a signal that is turned ON and OFF quickly because of the response time of your eye. The motors cannot start and stop instantaneously so a similar method can be used to control their speed. Instead of the averaging happening in your eye it happens in the energy exchange in the motor itself.

**Question 18**: Describe using the LED as an example how you would use this method to control the motor of the cars. You will experiment with this method – commonly referred to as Pulse Width Modulation - in a future lab.