WORK PACKAGE 6: SELECTIVE PROJECTS

In this work package, you will work on two "real-world" mini projects. You must solve Exercise 1 (DC Motor controller), and in addition either Exercise 2 (Parking Assistance System) or Exercise 3 (Stargate Lights).

EXERCISE 1: DC MOTOR CONTROLLER

Your first task is to develop a position controller for a DC motor. A DC motor converts direct current into rotational energy through a rotating magnetic field. Many DC motors are equipped with *Encoders* which track the position of the magnetic field, and thus the rotation.

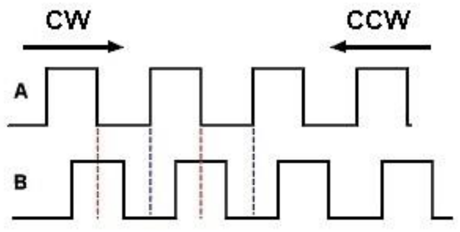

FIGURE 1: ENCODER SIGNAL WHEN MOTOR IS ROTATING CW: CLOCKWISE ROTATION CCW: COUNTER-CLOCKWISE ROTATION

An Encoder provides two signals: A and B. As you can see in Figure 1, when the motor is rotating clockwise, the **rising flank** of Signal A always precedes the rising flank of Signal B. The opposite happens (Signal B precedes Signal A) when the motor rotates counter clockwise. Each full rotation of the motor causes a fixed number of rising flanks for the two signals.

Your task is to control the position of the DC motor (in degrees) using the encoder signals as input.

To setup a prototype for developing the controller, use the following materials:

TABLE 1: BILL OF MATERIAL

Component	**Description**
DC Motor with Encoder	Use the one with the white connector. Do not use the one with black connector. **SET THE RPM to 45 (click on the motor, and there you can adjust the RPM).**
Power Supply	Set to 12 V and 0.6 A.
H-bridge Motor Driver (L293D)	https://en.wikipedia.org/wiki/H-bridge
Arduino	Connect the Encoder Signal to IO Ports 2 and 3, and Input 1 and Input 2 to IO Ports 5 and 6. See Figure below. Don't use other ports.

The detailed setup is illustrated in Figure 2. Pay attention to connect everything EXACTLY how it is illustrated in the Figure.

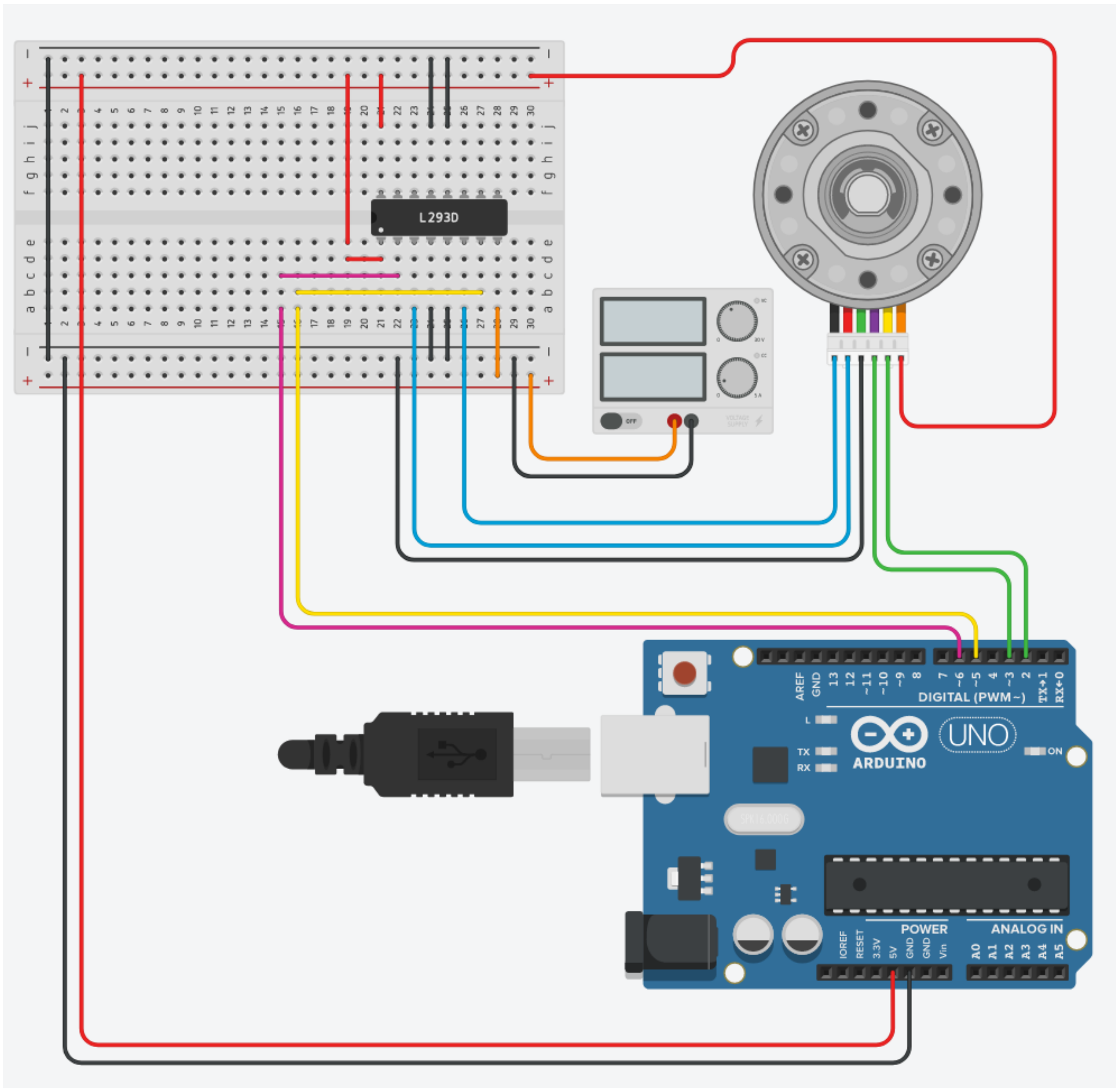

FIGURE 2: SETUP OF THE PROTOTYPE

Task 1: Driving the Motor

Your first task is to write a software that allows the Arduino to control the speed and direction of the DC motor. The program shall receive speed and direction through a serial (keyboard) input.

Task 2: Reading the Encoder

Your second task is to write an interrupt routine that reads the two encoder signals (connected to IO ports 2 and 3). Conduct a small experiment to figure out how many rising flanks of Signal A and Signal B a full rotation of the motor cause. Use this information to update the position of the rotor (in degrees) and the direction of rotation while the motor is spinning. Output the current position in degrees of the rotor on the serial monitor.

TASK 3: A POSITION CONTROLLER

Your final task is to develop a controller routine as shown in Figure 3, that allows you to drive the motor to a specified position (in degrees).

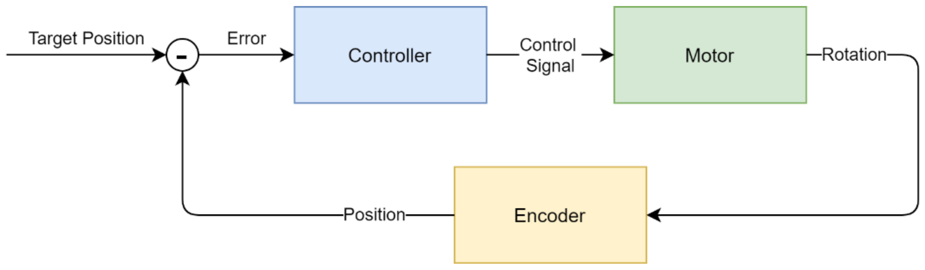

FIGURE 3: CONTROLLER STRUCTURE FOR A POSITION CONTROLLER

- The program should receive a desired rotor position in degrees through the serial terminal (e.g., 270 degrees).
- Then, based on your current position (assume always that when you switch the system on, the rotor is at 0 degrees) you need to calculate an error term between the current and the desired position. Output the error on the serial monitor.
    - $e = pos_{is} - pos_{desired}$
- Finally, use the error term to derive a control signal u that is fed into the motor. The control signal can be calculated as
    - $u = K_p \cdot e$
- Tune K_p to achieve a nice performance.

EXERCISE 2: PARKING ASSISTANCE SYSTEM

You started a small start-up specialised in ultrasonic parking assistance systems for vehicles. A major Swedish truck company contacts you because too many truck drivers cause parking accidents when reversing into a parking spot. Therefore, they want you to design a new parking assistance system.

The new parking assistance system contains an ultrasonic system as sensor, a multipurpose processing unit, a head-up display with 4 red LEDs, and a speaker.

The requirements for the system are as follows:

- The system shall be able to detect an obstacle behind the vehicle with a maximum distance of 200 cm and a minimum distance of 25 cm.
- The LEDs shall indicate how close an object is to the vehicle, where all 4 LEDs shall be lighted when the obstacle is closer than 30 cm and the first LED shall light when the object is closer than 200 cm.
- A tone from the speaker shall indicate how close an object is. The tone shall be activated when the object is closer than 200 cm. The closer the object gets to the vehicle, the more "annoying" the tone shall become.
- If the object is closer than 25 cm, all four LEDs shall blink, and the tone should become distinctively super annoying to indicate the immediate danger.
- For better maintainability, each feature (i.e., UltraSonic distance detection, LED lights, and Speaker control) shall be controlled through individual functions.

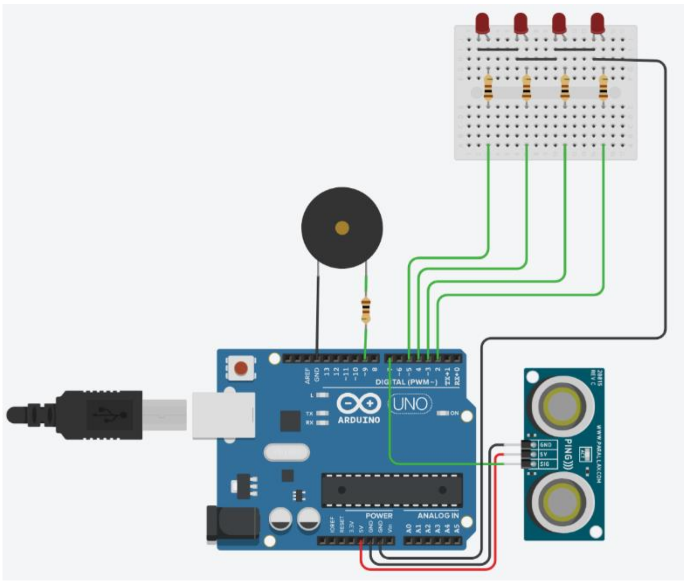

FIGURE 4: PROTOTYPE DESIGN FOR AN ULTRASONIC BASED PARKING ASSISTANCE SYSTEM

To decide if they want to hire your company for this project, they ask you to prepare a small prototype demonstrating the concept. Your "hardware guy" already prepared a schematic of the prototype, see Figure 4.

Furthermore, your "hardware guy" reminds you that an ultrasonic sensor works by first sending out a ping and then to switch to "listening" and to wait until an echo returns. She/he says something like this could work:

- Set Pin to Output.
- Write a zero to the output to clear the system. Wait 2 milliseconds.
- Write a one to the output to send a ping. Wait for 10 milliseconds.
- Write a zero to the output to stop sending the ping.
- Set Pin to Input.
- Listen to input if some echo returns. The time until the echo returns is related to the distance of the object from which the ping reflected.

Your task is to build the system according to the proposed schematic (you can choose if you want to use "real components" or a virtual setup on TinkerCad) and to develop the necessary software.

TABLE 2: BILL OF MATERIALS FOR PROJECT 2

Component	**Description**
Ultrasonic Distance Sensor with 3 PINs	Use the one with 3 pins, not the one with 4 pins. Connect Power and GND to 5V and GND of the Arduino. Connect Sig to IO Pin 7 of the Arduino.
Piezo (Speaker)	Connect to IO/Pin 9 via a 100 Ohm resistance and to GND.
Mini Breadboard	
4x RED LEDs	Connect the 4 LEDs, through 100 Ohm resistors, to IO Pins 2-5 of the Arduino.
5x 100 Ohm resistors	See above
Arduino Uno	

Exercise 3: Stargate

You started a small start-up company specialised in software for light effects on film sets. A major Hollywood studio producing a new franchise of the popular sci-fi series "Stargate" contacts you, because they want new cool light effects when the Stargate is dialling out to other planets. To achieve the light effect, the Stargate is equipped with 24 individually addressable RGB LEDs as depicted in Figure 5. A Stargate address contains 7 digits, and only a handful of addresses lead to a successful Stargate connection between planets. A list of correct addresses will be provided.

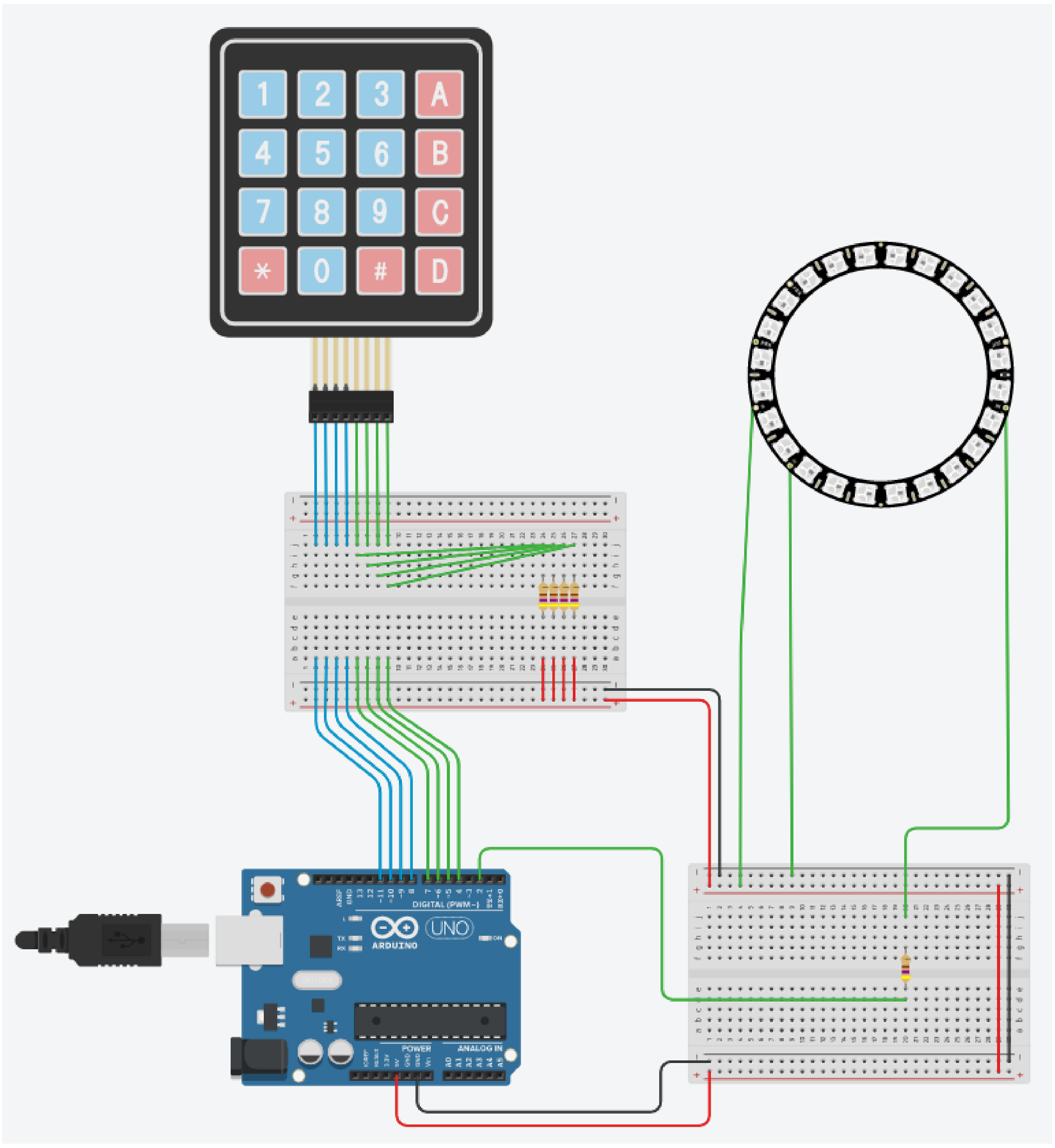

FIGURE 5: SETUP OF LIGHTNING SYSTEM FOR THE STARGATE

Your task is to setup a prototype as shown in Figure 5. Create a program, such that the following requirements are fulfilled:

- A Stargate address contains 7 digits. All 7 digits are entered at once.

- As soon as all seven digits are entered, the Stargate starts dialing. Create a fancy light effect that shows that the gate is dialing[1]. For example, let a light spin around the LED ring, changing colour, or any other fancy animation you can think of.
- After a few seconds (2-3), the first digit “locks into” the Stargate. Locking in means that two LEDs start lighting constantly, indicating that the Stargate accepted that digit (see Figure 6).
- The dialing animation continues for the second digit. Again after 2-3 seconds, the second digit “locks into” the Stargate. Now four LEDs at the respective positions light constantly.
- Dialing and locking in continues until 6 digits are locked into the Stargate.
- The seventh digit (two LEDs on top of the Stargate) only lock in, if the entered address is in the list of valid addresses.
- If the entered address was invalid, an “error animation” (like all LEDs blinking red or something other cool you can think of) occurs instead of locking in the 7th digit.

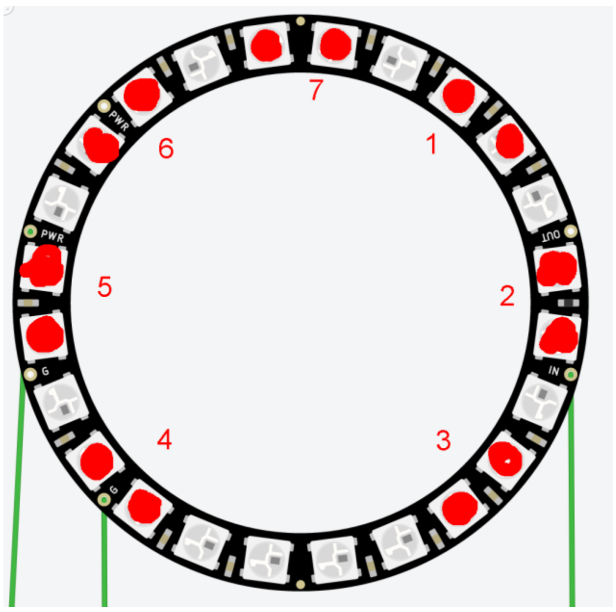

FIGURE 6: LEDS INDICATING SUCCESSFUL "LOCKING IN" OF ADDRESS DIGITS

TABLE 3: BILL OF MATERIALS FOR PROJECT 3

Component	**Description**
Keypad 4x4	Connected to IO/Pins 4-11 of the Arduino, similar to the previous Work Packages.
NeoPixel Ring 24	Connect, through a 470 Ohm resistance, to Arduino IO port 2. Connect to 5 V power.
Arduino	
5x 470 Ohm resistors	4 are used to prevent bounce back of Keypad, one is used to protect the circuitry of the Neopixel ring.
Arduino Uno	

[1] See from 00:20 https://www.youtube.com/watch?v=24KP9czci8s

Exercise 4: Pulse oximeter

In this exercise, your task is to develop a pulse oximeter and visualize the result based on the multimeter and on the serial port.
You can use the following tutorial for the schematics of such a device: Interfacing MAX30100 Pulse Oximeter Sensor with Arduino (how2electronics.com)
You can borrow the necessary components from the course responsible (I have one sensor) during the course. First come, first served).