THP – Electronics

By Krutarth Mehta (24B0055)

Q1 – Understanding embedded systems in Arduino

Answer –

For any general use, there are many sensors available in TinkerCAD. There are switches, temperature sensors, moisture sensors, flex sensors, light sensors, etc, to name a few. However, for the given task, only some have the potential to be used effectively. I found that a slide switch worked best when providing input because it is easy for users to give it – they have to flip the switch.

Circuit and code explanation

Four slide switches are used to give the input, one switch for each digit. In the OFF position, no voltage is detected by the digital pins of Arduino, while in the ON position, high voltage is detected by Arduino. Taking high voltage as 1, we can get the 4 bits of the binary number. After converting the binary number to its decimal equivalent with a simple equation, Arduino outputs the decimal number through the serial monitor.

TinkerCAD circuit design

https://www.tinkercad.com/things/1Ey6xJf8Cdt-q1electronicsthp?sharecode=AtgE-oSwyagnovWx5MEO94eYlQ2Elu0i_L5KqObeTR8

Q2 – Programming a Humanoid's Wheelbase for Movement

Answer –

{I have assumed the humanoid takes 500ms to turn 90 degrees in any direction.}

Code explanation

I have distributed the path into two arrays, both consisting of either right or left turns only. The elements in those arrays are the path lengths of each path in units of how many motor revolutions are required.

I used one for loop for each array, which would make the bot go straight for a given time, then turn right or left, depending upon which for loop is being executed. As the elements of arrays are traversed by the for loop, the duration that the humanoid goes straight changes.

The motors are running at 231rpm (max) for the given circuit in the simulation. So,

time taken for humanoid to complete five motor revolutions = 5*60*1000/231 = 1298.7ms

Similarly, time for 10 and 15 motor revolutions can also be calculated.

I have used a variable called wait in both for loops to track how long the humanoid should go straight along each path.

After the completion of for loops, some code is present to make the bot turn left one more time by 90 degrees so the bot makes a full about-turn at point D.

TinkerCAD circuit design

https://www.tinkercad.com/things/gMoGmGehjyQ-q2electronicsthp?sharecode=4G311Z1VvjPQbEIcTGBJ4H9zq5QBDf0-ZEcpqHGbEEs

Bonus task

- **Components used**
  1) Arduino Uno R3
     Datasheet - https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf
     Price = Rs. 2671 (29.30 EUR) { https://store.arduino.cc/products/arduino-uno-rev3 }
  2) L293D Motor driver
     Datasheet - https://www.ti.com/cn/lit/ds/symlink/l293d.pdf?ts=1729457116521
     Price = Rs. 145 (1.73 USD) { https://www.lcsc.com/product-detail/Gate-Drivers_Texas-Instruments-L293DNE_C701118.html }
  3) Power supply (9V battery used in this case)
     Price = Rs. 25 { https://robu.in/product/9v-original-hw-high-quality-battery-5pcs/ }
  4) Hobby Gearmotor
     Datasheet - https://resources.kitronik.co.uk/pdf/2547-yellow-right-angled-geared-hobby-motor-datasheet.pdf
     Price = Rs. 43 (0.51 USD) { https://srituhobby.com/product/5v-dc-gear-motor-for-smart-robots/ }

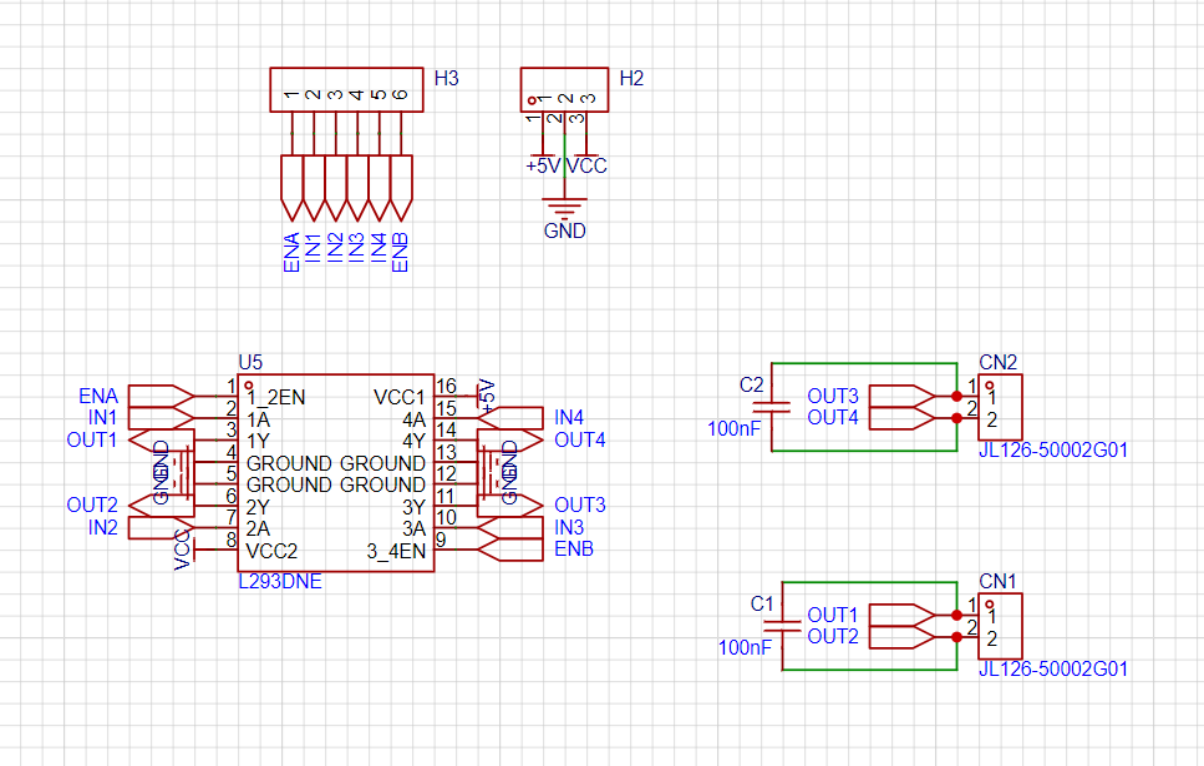
     Total cost = Rs. 2927

- **TinkerCAD limitations**

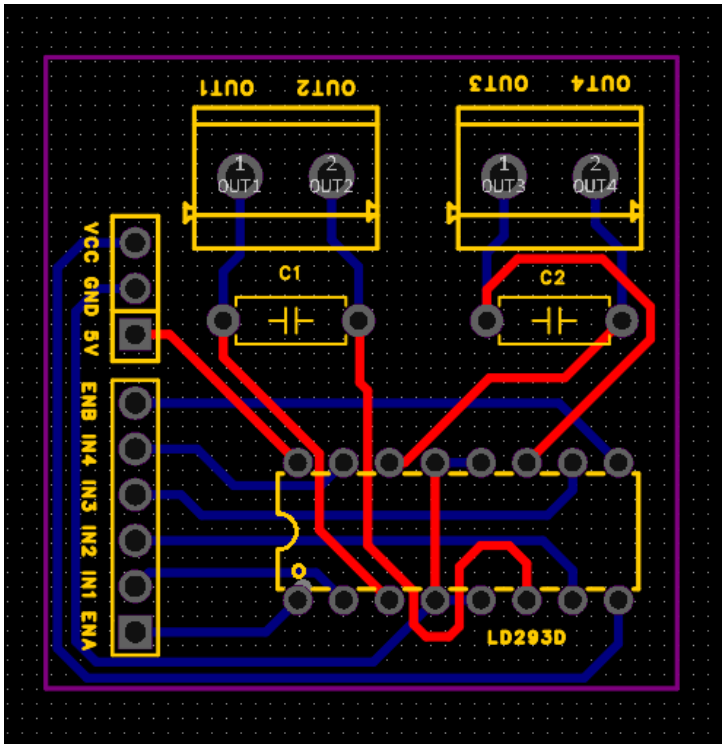
A capacitor has to be connected in parallel with the brushed DC motors to reduce noise. When the motor runs, the current changes too fast because of the constant connection and disconnection between the armature and power supply through the brush – the commutator connection. This creates magnetic interference and noise in nearby radio devices. A capacitor reduces the noise and smoothens the rotation of DC motor during changes in load.

Q3 – Designing a custom PCB for Humanoid's Wheelbase Breakout Board

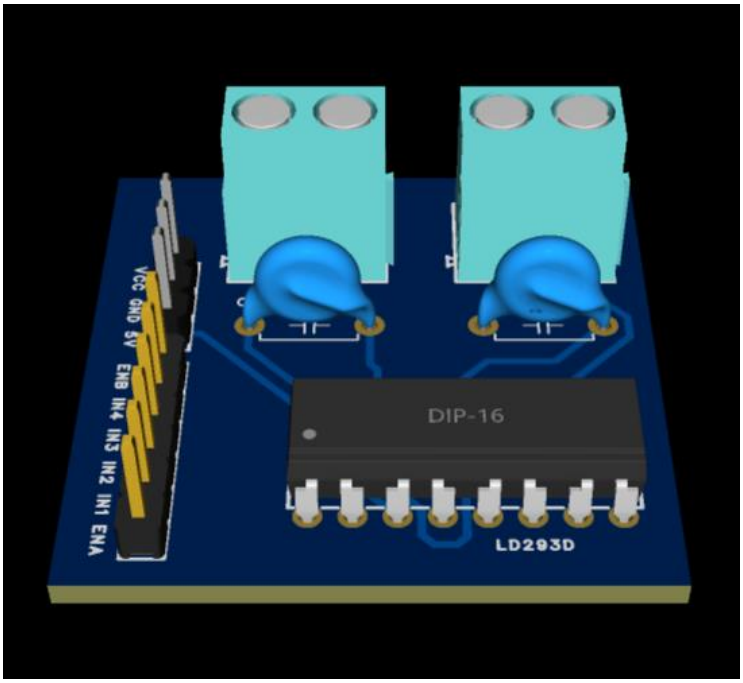
Schematic



PCB Routing



PCB in 3D



Q5 – Multi Arduino communication for humanoid control

Answer –

{I have used an ultrasonic distance sensor (4 pins) to measure distance instead of the UV sensor mentioned in the question}

I2C communication protocol

I2C (Inter-Integrated Circuit) protocol is a communication protocol that can be used to communicate between multiple Arduino boards, an Arduino board and a breakout board, etc. It consists of one controller device (master Arduino, for example) with one or more peripheral devices. These are connected to the controller's SCL (serial clock pin) and SDA (serial data) pins. Though each device is functionally independent from others, they still respond when prompted by the controller device.

During the communication, the Arduino controller device pulses at a regular interval through the SCL line. As the SCL line is connected from the controller device to all peripheral devices, all devices are in sync with the same clock.

As the clock line changes from low to high (rising edge of the clock pulse), one bit of information is sent through the SDA line. Similarly, more bits are sent until 7 – 8 bits comprise an address, and more bits constitute a command or data. This is sent bit by bit to the peripheral device according to the address, which executes the request and sends back a response if required.

There are different bits between data packs or between address and data packs for whether the controller device wants to read or write, as well as acknowledgement bits, which let the controller device know the receiver acknowledged the information.

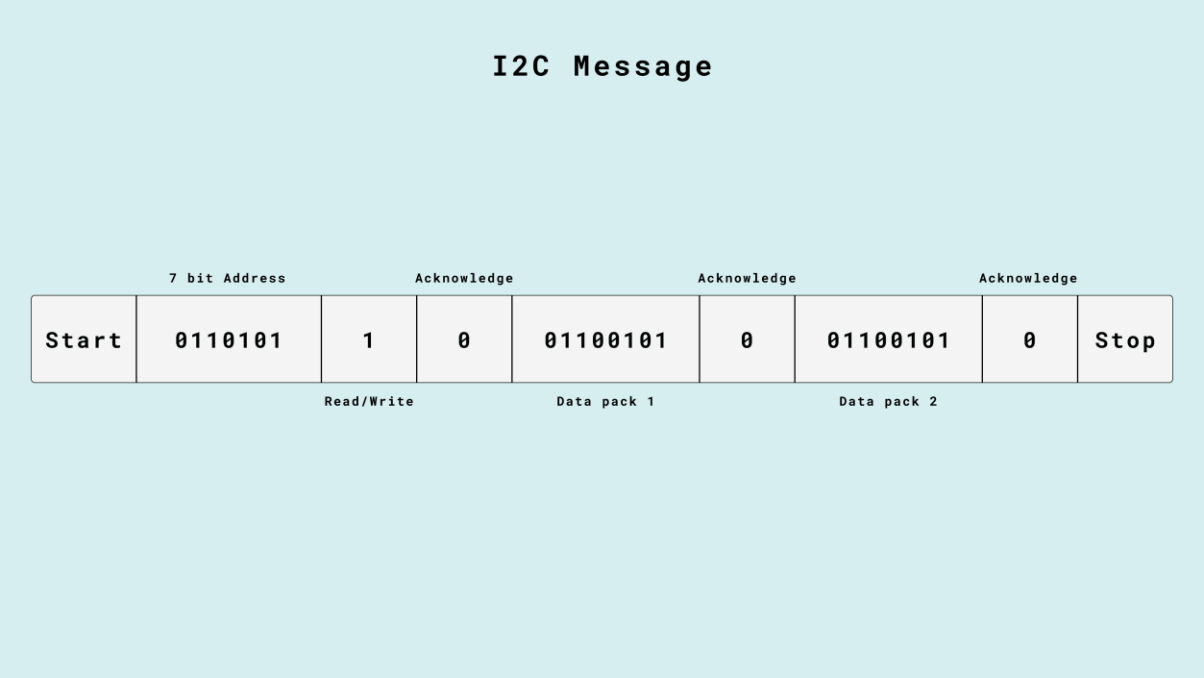


Image source - https://docs.arduino.cc/learn/communication/wire/

Code explanation

1) Master Arduino

The master will constantly request bytes from slave 1. It will check if the byte received is correct or not. If 0 is received as the byte, it is considered incorrect and is ignored. If 1 is received as byte, which is considered the correct byte, the if statement block will be executed, and it transmits another byte to slave 2.

2) Arduino Slave 1

The ultrasonic sensor sends a high-pitched sound wave whenever the code tells the sensor using the trigger pin. The time taken (in microseconds) for the wave to go and come back as an echo can be measured by pulse duration in the echo pin. To match the distance TinkerCAD gave in the ultrasonic sensor with my code, I used the speed of sound as 344.6 m/s.

Once the object is less than 15cm (of course, this threshold can be changed as required), a byte is sent to the master. At the start of the simulation, no byte is sent even after the

master requests it if the object isn't closer than 15cm. After the sensor detects an object closer than 15cm, the slave sends bytes whenever the master requests. An issue with this is that the slave keeps sending data even when the object is not within 15cm. To solve this, I used a variable called signal, which can take either 0 or 1 values. This variable's value is sent as a byte by the slave to its master whenever the master requests it. When the object is within 15cm, the signal takes value 1 and sends that byte.

3) Arduino Slave 2

This Arduino slave turns on the motor when no object is detected within 15cm. When an object is within 15cm, the master sends a byte to slave 2. This slave then executes the handler function of onReceive(), which turns off the motor. Wire.read() is essential as it reduces the number of bytes available to read so that Wire.available() returns 0 after execution.

After the handler function is executed, there has to be some way through which the regular code which turns on the motor is not executed. This is accomplished by a variable called cond. The default value of cond is 1. When the handler function is not executed, the code turns on the motor with cond=1. When the handler function is executed, the value of cond turns to 0, which does not execute the code for turning on the motor. At the end of loop(), the default value of cond is restored to 1.

TinkerCAD circuit design

https://www.tinkercad.com/things/8ui885NdVwA-g5electronicsthp?sharecode=RxiVLMLCEXk2v_gzLVwgvtGa-DAAiPaa3y3tDUuNcUE

My Github Repository for all codes and other files:

https://github.com/BusyPear/THP-Electronics-code-by-24B0055.git

Resources Used

General -

https://docs.arduino.cc/language-reference/

Q2 -

https://electronics.stackexchange.com/questions/490934/why-do-we-use-capacitors-in-parallel-with-dc-motors

https://byjus.com/question-answer/why-capacitor-is-used-in-dc-motor/

Q3 –

https://soldered.com/learn/breakout-boards-what-are-they-and-why-you-should-use-them/

https://www.programmingelectronics.com/what-is-a-breakout-board-for-arduino/

https://jlcpcb.com/blog/understanding-and-utilizing-breakout-boards-in-electronics-projects

https://www.flyrobo.in/l293d-motor-driver-stepper-motor-driver-module-for-arduino-and-diy-projects?tracking=ads (inspiration)

https://www.instructables.com/L293D-Motor-Driver/

Q5 -

https://docs.arduino.cc/learn/communication/wire/

https://www.gammon.com.au/i2c

https://docs.arduino.cc/language-reference/en/functions/communication/Wire/