Simon Says Project Walkthrough

1. Setup

To start using the EV3 with Alexa we require

a. Configure the development environment, we will use Visual Studio Code
b. Install the ev3dev software

A detailed description of these steps can be found in the web site
https://www.hackster.io/alexagadgets/lego-mindstorms-voice-challenge-setup-17300f

2. Register an Alexa Gadget

Once that the setup process has been completed the next step is the registration of an Alexa gadget, in our case the EV3. The registration can be made on the web site https://developer.amazon.com/. More details can be found at https://www.hackster.io/alexagadgets/lego-mindstorms-voice-challenge-mission-1-f31925.

3. Connect the EV3 to Alexa

The EV3 is added to the Alexa device as a Bluetooth device as is shown in https://www.hackster.io/alexagadgets/lego-mindstorms-voice-challenge-mission-1-f31925.

4. Test the EV3 and Alexa connection

This step is used to verify the connection between Alexa and the EV3. In the test, the EV3 lights will change and it also will make a sound when Alexa detects the wake-up word. A detailed description of these steps can be found at
https://www.hackster.io/alexagadgets/lego-mindstorms-voice-challenge-mission-1-f31925

5. Create Alexa Skill

The current project requires the creation of a new skill. Sign in to
developer.amazon.com.

Once that you are logged, go to Alexa console and create a new skill following the steps of
https://www.hackster.io/alexagadgets/lego-mindstorms-voice-challenge-mission-3-4ed812

a. Add a new intent to the skill and call it StartGameIntent.
b. Add sample utterances
    a. new game

b. start game
c. start

6. Alexa lambda function

The Alexa lambda code is based on the example https://www.hackster.io/alexagadgets/lego-mindstorms-voice-challenge-mission-3-4ed812.

Add a new handler for the StartGameIntent and call it StartGameIntentHandler.

The StartGameIntentHandler function will generate a new sequence of colors, say the colors through the Alexa device and send the sequence to the EV3.

```
var speakOutput = 'Simon says, press ';

var s= Util.generate_sequence(1);

var str= Util.sequence2colors(s);
speakOutput += str;

Util.putSessionAttribute(handlerInput, 'color', str);

var mainTxt= 'Record '+  gameRecord + '<br>Score: ' + score + '<br>Level: '
+ level;
Util.putAPLmessage(handlerInput, 'Simon Says', mainTxt, 'To start a new game
 just say start game');

// Construct directive with the payload containing the sequence parameters
const directive1 = EV3.build(attributes.endpointId, NAMESPACE, NAME_CONTROL,
    {
        type: 'init',
        sequence: s
    });


return handlerInput.responseBuilder
    .speak(speakOutput)
    .addDirective(directive1)
    .getResponse();
```

In the EV3 the Alexa directives will be handled with the following function

```
    def on_custom_mindstorms_gadget_control(self, directive):
        """
```

```
        Handles the Custom.Mindstorms.Gadget control directive.
        :param directive: the custom directive with the matching namespace a
nd name
        """
        try:
            payload = json.loads(directive.payload.decode("utf-8"))
            print("Control payload: {}".format(payload), file=sys.stderr)
            control_type = payload["type"]
            print('control type: ',control_type)

            if control_type == "init":
                print('****************************************')

                self.sequence= payload["sequence"]
                print('New sequence received: ', end =" ")
                print(self.sequence)

                c= self.colores_secuencia(self.sequence)
                self.playing= True
```

As we can observe in the previous function the sequence of colors is received as a payload of the Alexa directive. Once that the sequence was received the function set the variable self.playing to true. This variable will activate the thread that reads the touch sensors as it can be seen in the following code.

```
    while True:
        while self.playing:
            c=-1
            #detect push
            while c == -1:
                c= self.color_sensor()
                sleep(0.01)

            t= self.color2sound(c)

            self.sound.play_tone(t, 0.1, play_type=1)

            #detect release
            c2=0
            while c2 != -1:
                c2= self.color_sensor()
                sleep(0.01)
            if self.sequence[i] != c:
                print('lose ')
```

```
                self.playing= False
                self._send_event(EventName.GAME, {'win': 0})
                i=-1

            elif i== len(self.sequence) -1:
                print('correct!!!')
                self.playing= False
                self._send_event(EventName.GAME, {'win': 1})
                i=-1

            i= i+1
        time.sleep(1)
```

As we can observe the code will check if the touch sensors were press as the sequence sent by the Alexa device. The function sends a 0 or 1 to the Alexa skill, when the sequence is incorrect or correct respectively.

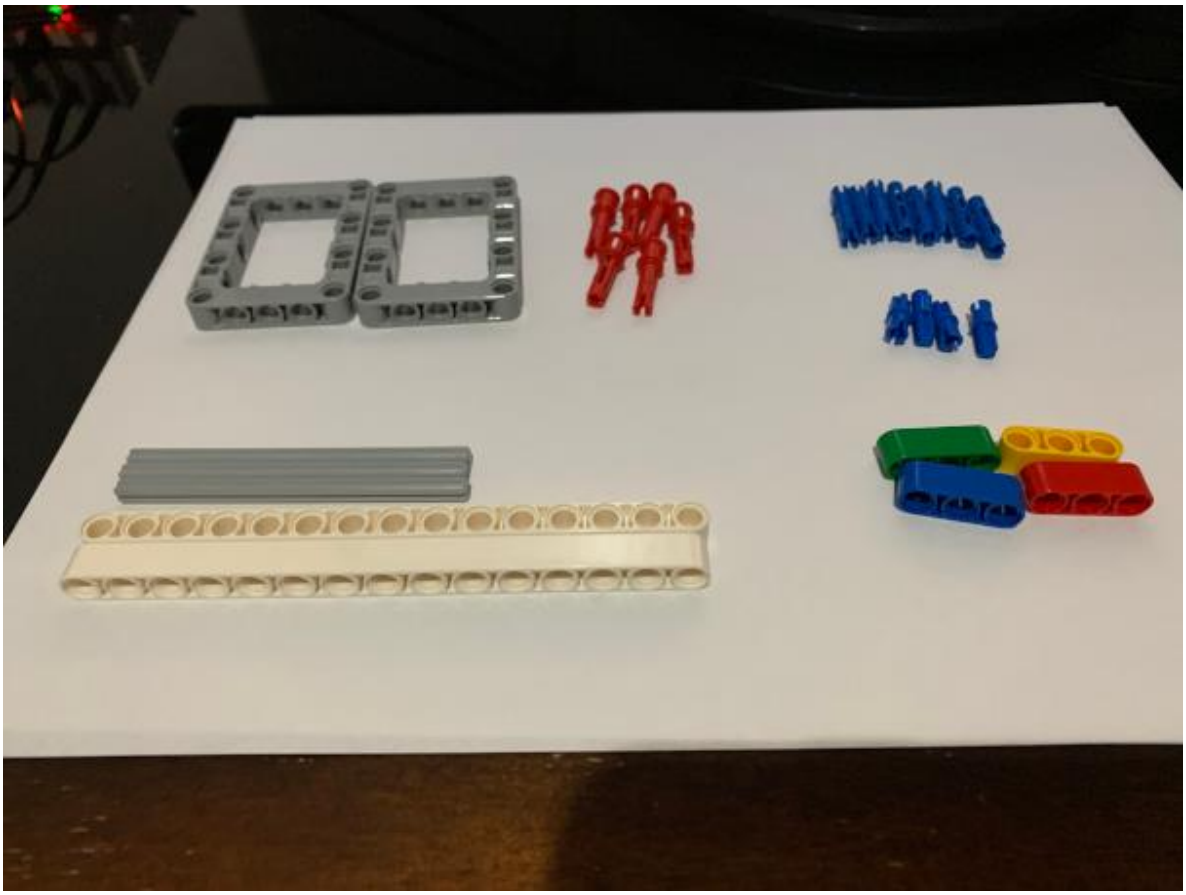
The answer sent by the EV3 is received in the Alexa skill handler EventsReceivedRequestHandler. The handler checks the payload of the event, in our case “win”. If this value is 1 we assume that the sequence provided by the user is correct then the score is increased, and we continue with the game. If the answer is 0 then the user sequence is incorrect and the game is over, and we inform the user his score.

Once the game is over the user will be asked if he wants to start a new game, if that is the case then the startIntent is called again and the process starts again.

The game record is saved using a S3 persistence storage. For more information about persistence please consult https://developer.amazon.com/en-US/docs/alexa/custom-skills/manage-skill-session-and-session-attributes.html#save-data-between-sessions

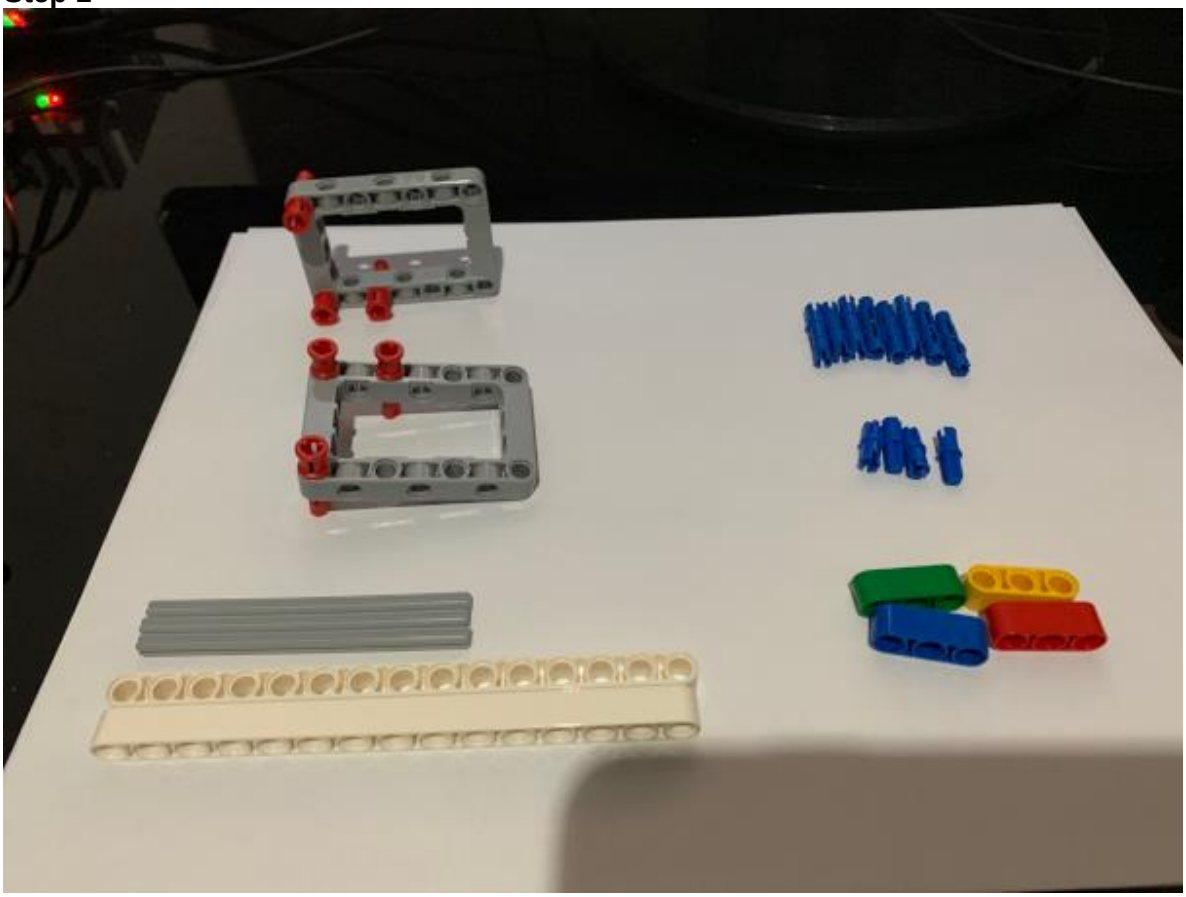
7. Build instructions

To build this project we require four touch sensors and the Lego components shown in the following image:

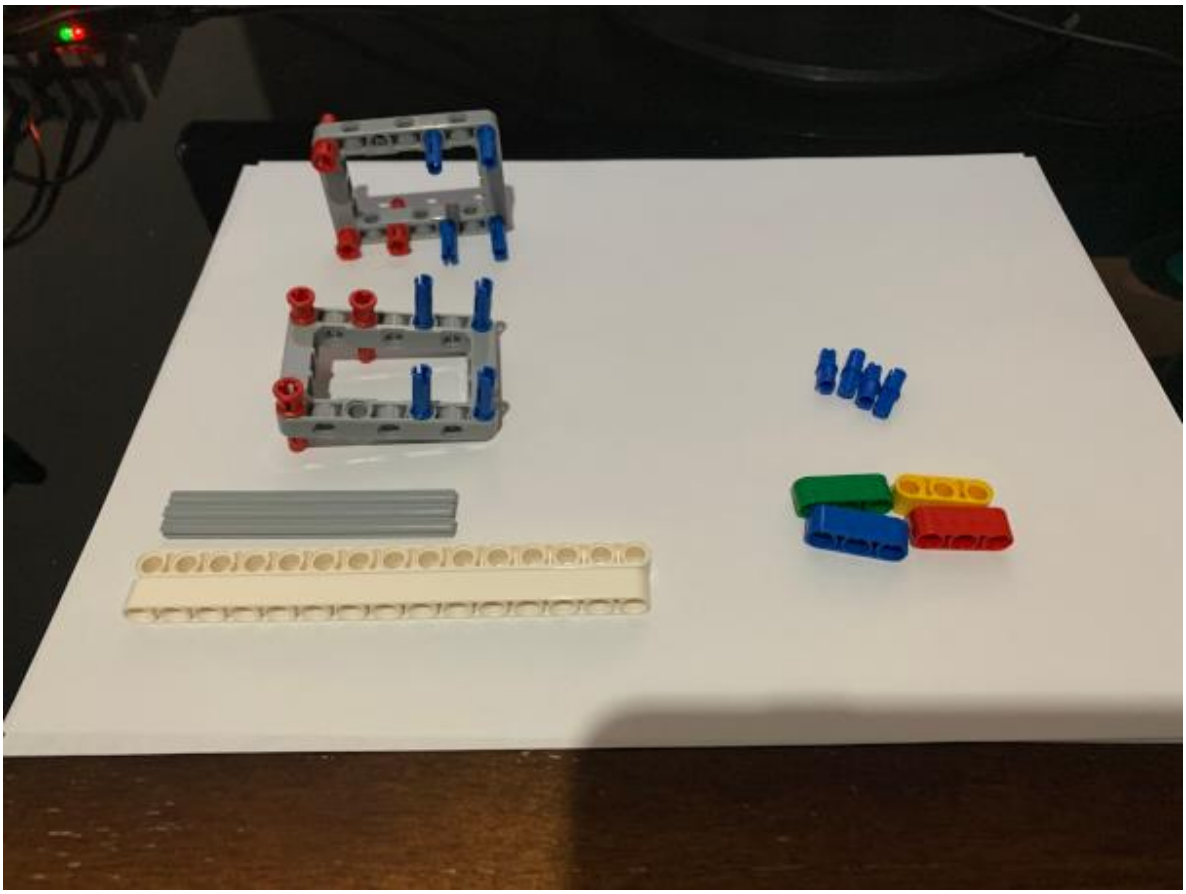


The steps are the following

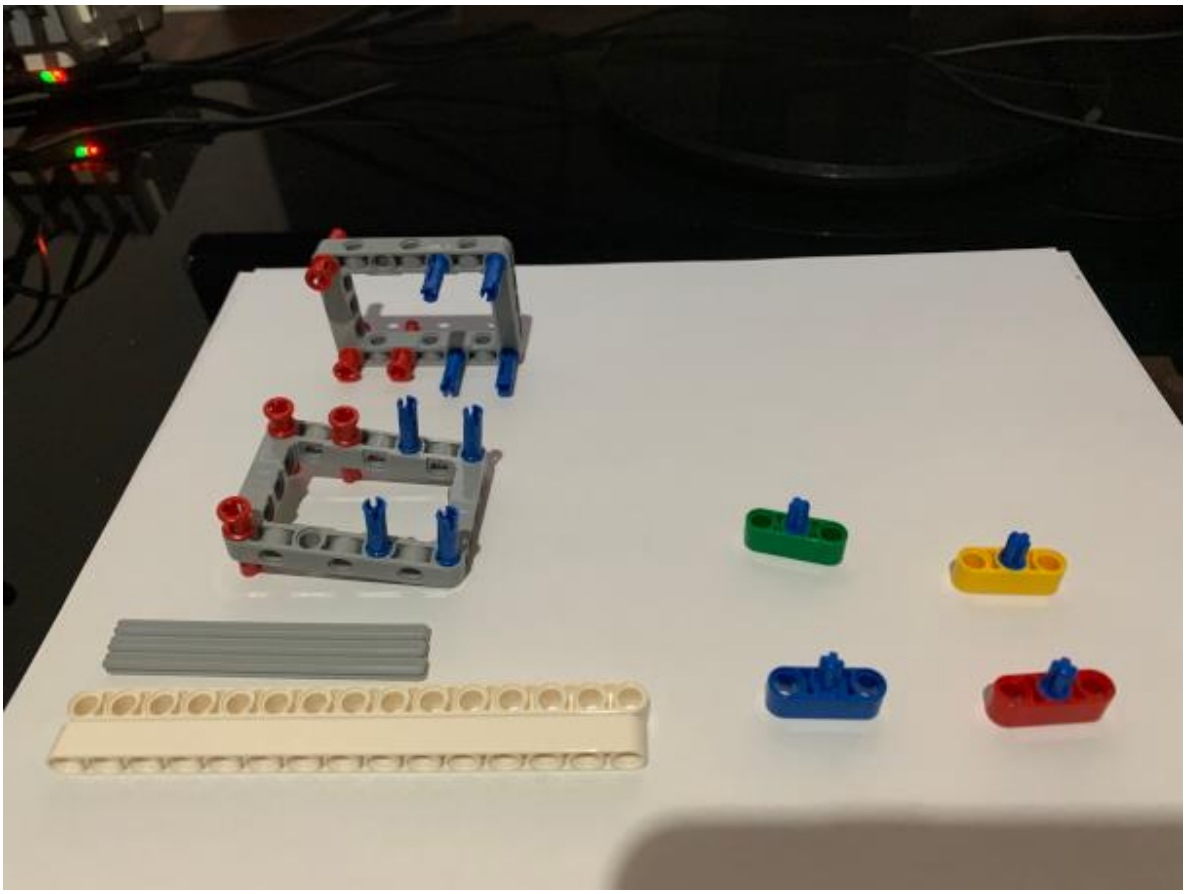
Step 1



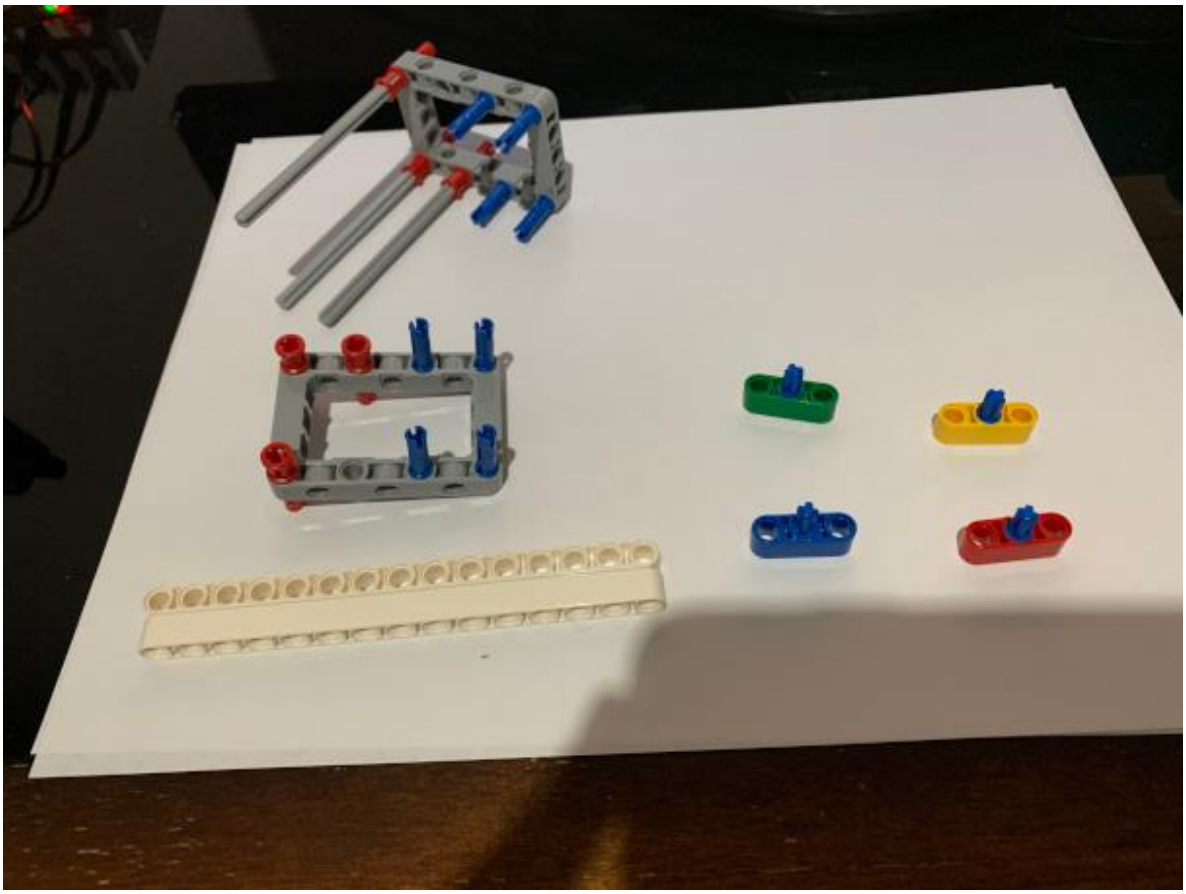
Step 2



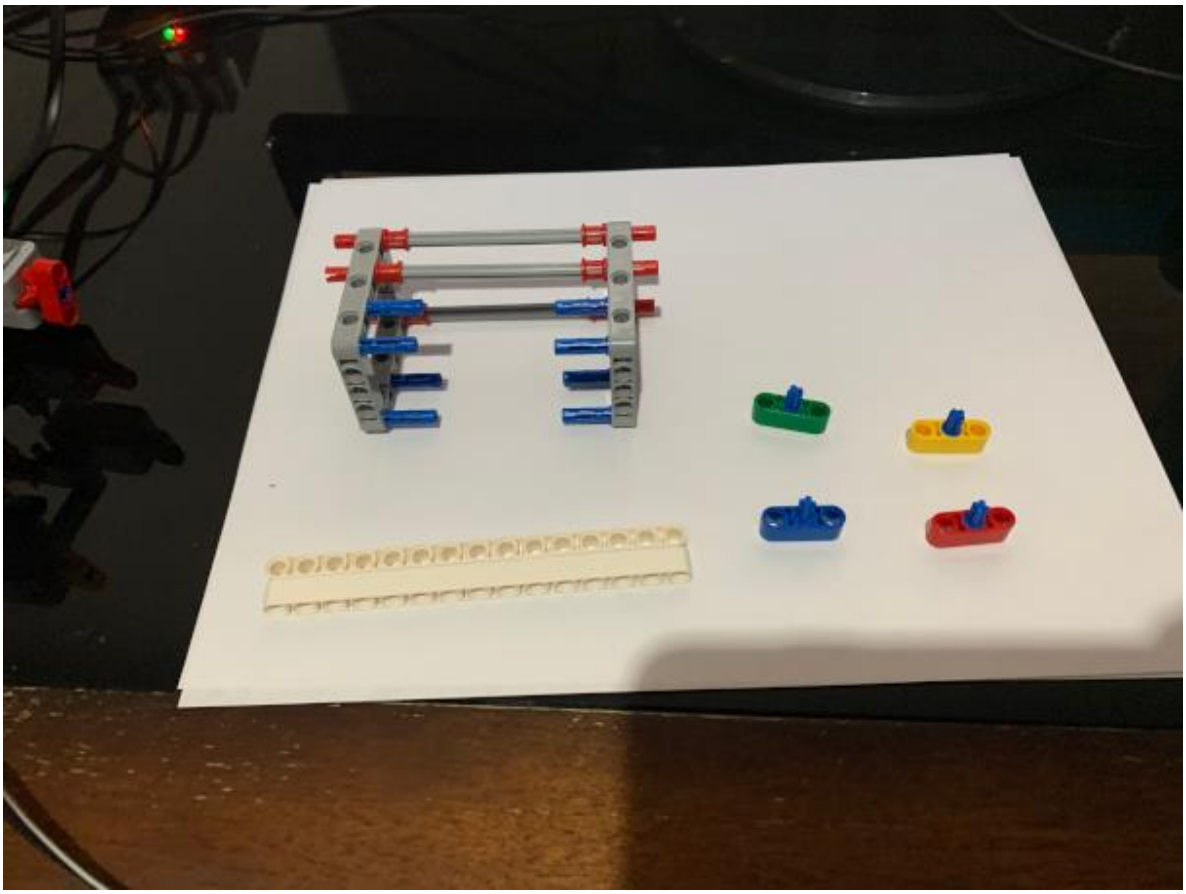
Step 3



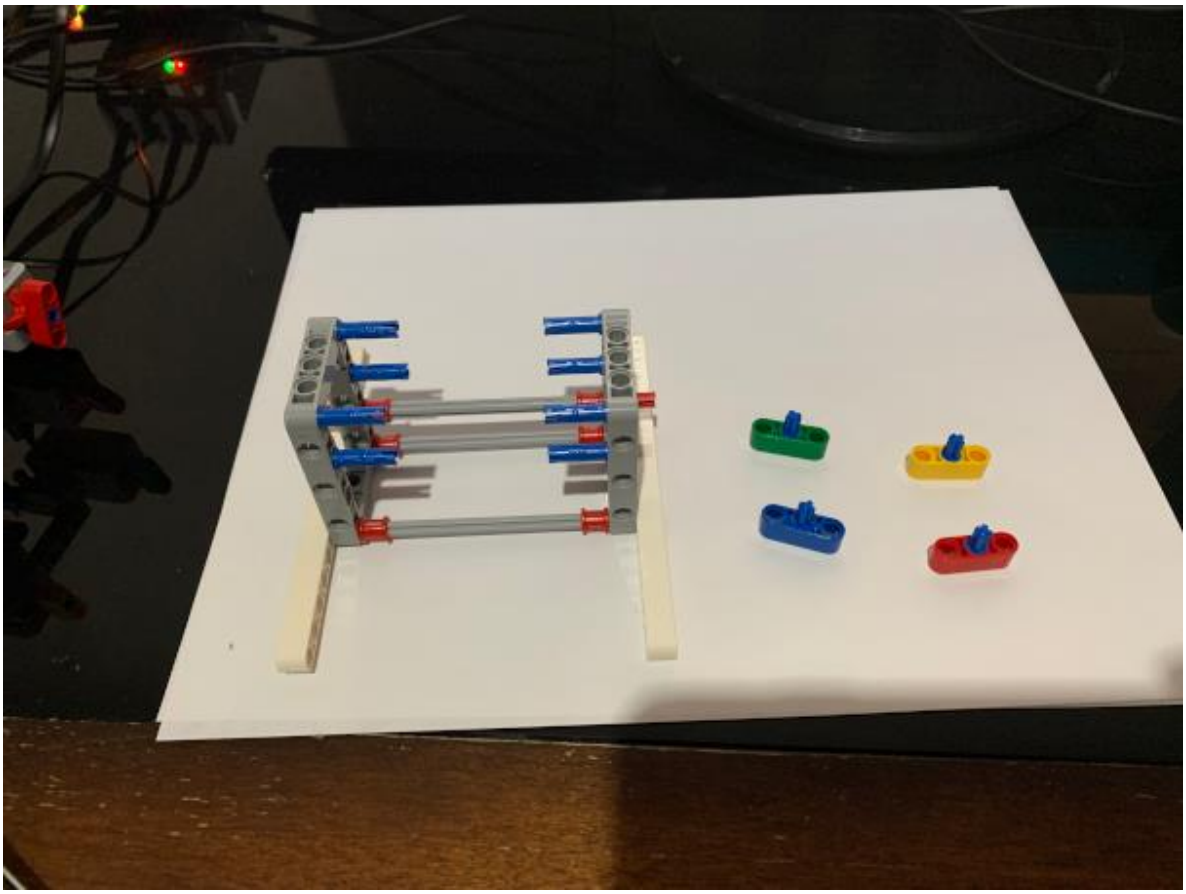
Step 4



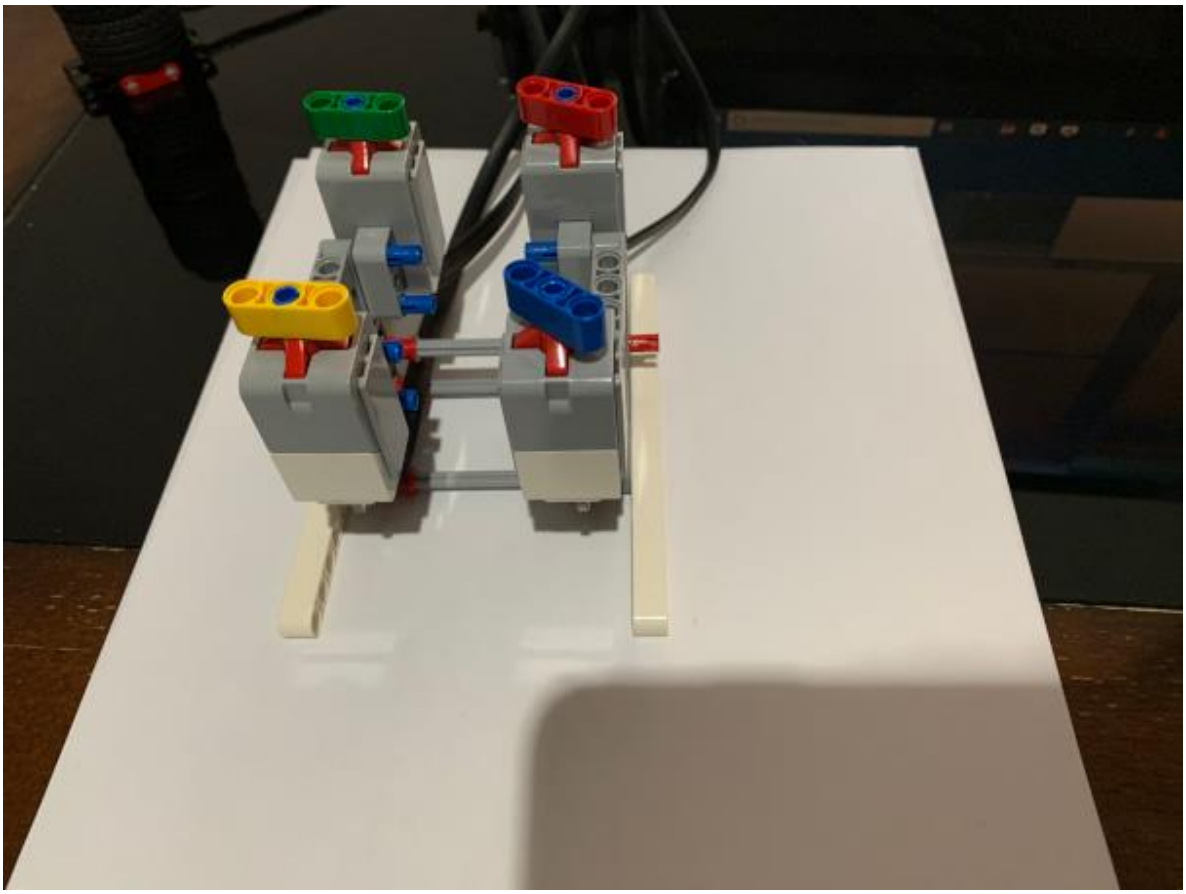
Step 5



Step 6



Step 7



Connections

The green button is connected to port 1.
The red button is connected to port 2.
The yellow button is connected to port 3.
The blue button is connected to port 4.