XLR8 - Digital Control - Report

For the Digital control of the Bot, the following controls were needed:

1. An Application that provides a good user interface for controlling the bot's direction of motion and speed and is also capable of connecting to the esp32 via Wifi.
2. Esp32 libraries which could be used to connect this app to the esp32's wifi hotspot and in turn, send signals to control the bot.
3. Implementation of these signals using functions in Arduino

MIT APP INVENTOR

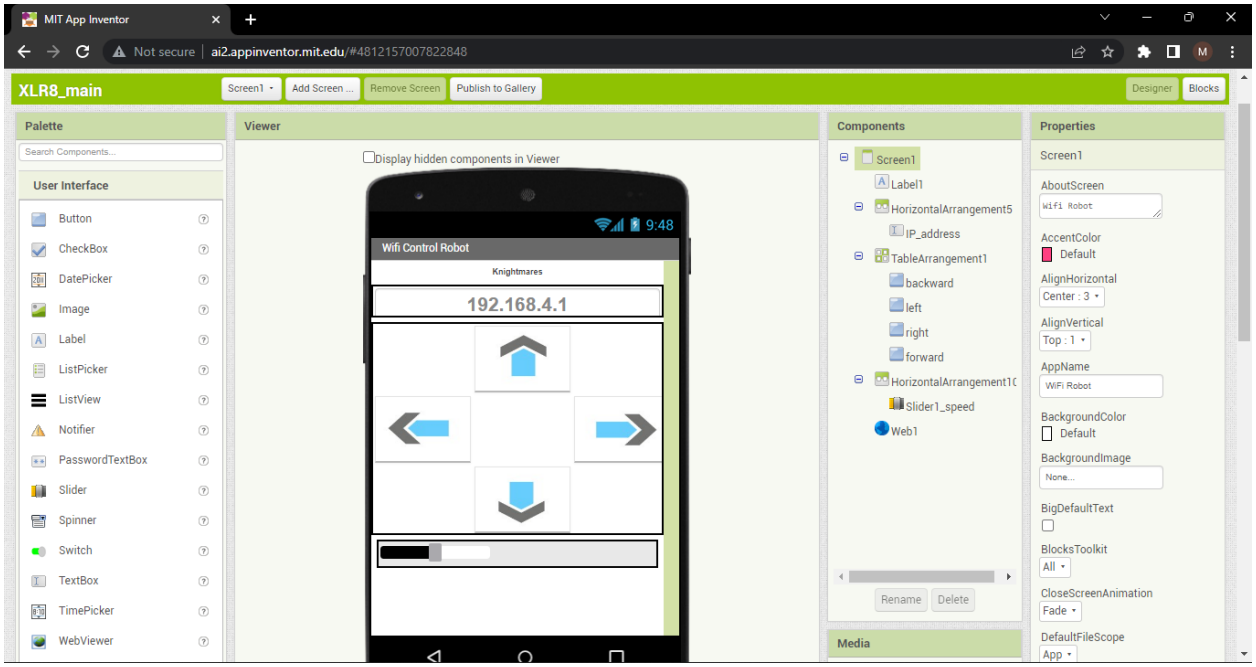
A website designed by MIT to create any app along with its specific user interface and code being implemented in the form of blocks.

User Interface:

Since we are controlling the bot's direction and speed, we require 4 buttons for each direction and a slider whose input value gives the command for the speed to the bot. The IP address also was added on the top to connect to the specific IP address of the bot, if needed.

Components were added using the list in the left column of the screen and they were customized with pictures of the arrows as shown below. The slider was added below and the input values taken were from 1 to 10, with 1 being slider at the leftmost point and 10 being slider at rightmost point.

In the rightmost column of the screen, Properties of the app like app name, background color etc, were specified.

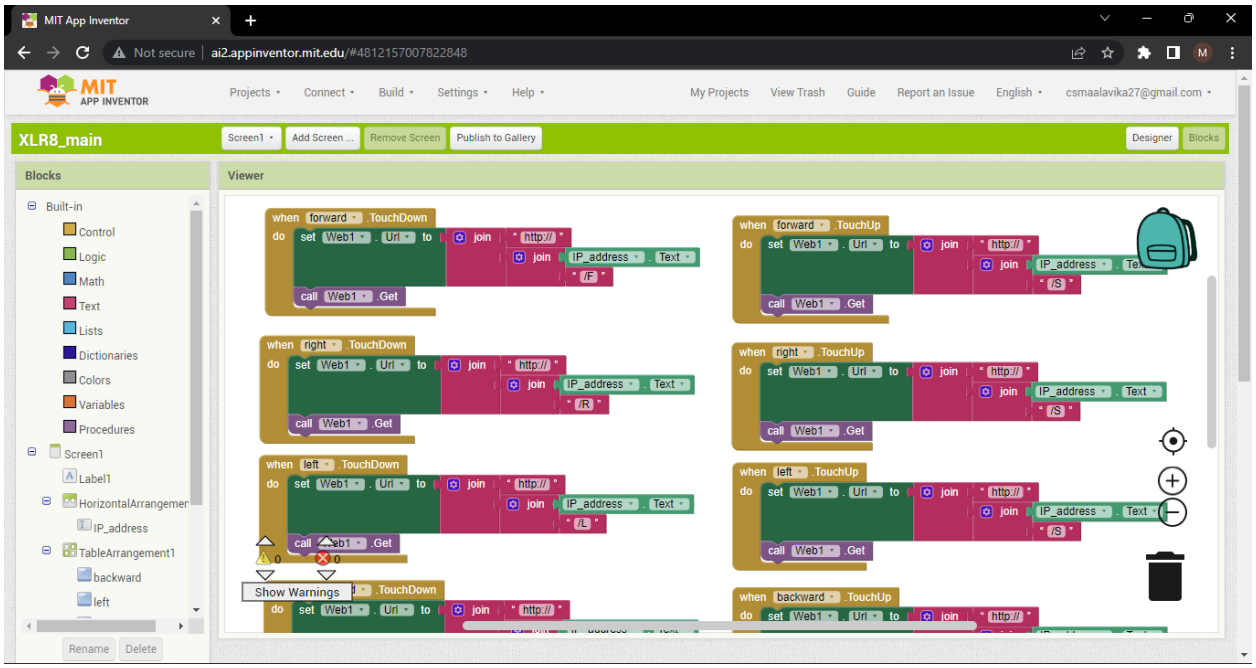


Logic:

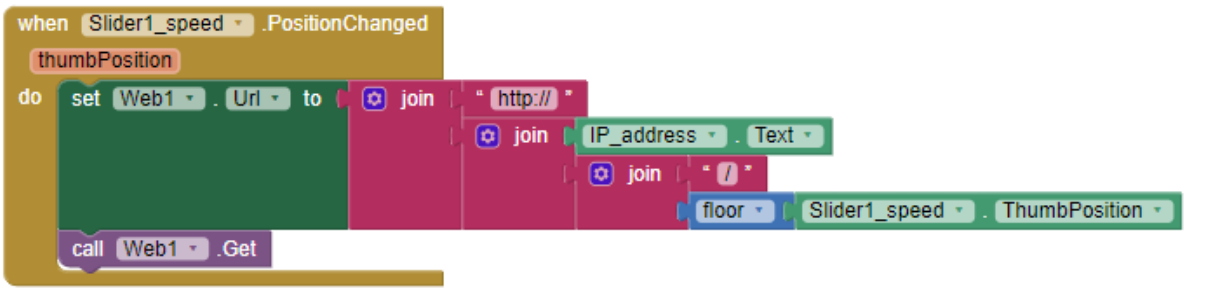
Each button, when clicked, or each value input from the slider (1-10) was directed to a specific URL of the form “http://<IP address given in the bar>/<command>.

Here, the IP address has been specified as 192.169.4.1. The command was given as ‘L’ for left, ‘F’ for forward, ‘R’ for right,‘B’ for backward and the corresponding value 1-10 for the slider input. Each button sends a message to esp32 in the form of which URL it gets directed to.

In the below image, blocks have been used to implement this logic. When the button is clicked, the corresponding URL is visited *in the background*. And when the button is unclicked, a null URL with command ‘S’ is visited.



Below are the blocks for getting slider inputs. The command sent in the URL is the floor of the slider position in order to get an exact integer between 1 and 10.



App creation:
To create the app, click on Build and then Android App. After some processing, the QR Code and link to download the app is obtained. You can download this app, connect to the esp32's wifi hotspot and use it easily to control the bot.

ESP32 WIFI CONNECTION:
The above explains how the app can be used to create a user interface for the operator and also, how to send information to the esp32. Now, we shall look into how we use Arduino IDE to extract these messages to the esp32.

```
#include "WiFi.h"                          // including the Wifi
library
#include "ESPAsyncWebServer.h"     // including the Web server
library
```

These libraries need to be installed initially and then included into the code.

Wifi.h module - Enables network connection (local and Internet).
With this library you can instantiate Servers and IP addresses can be assigned statically. This is the base library that is a necessity to create server functions.

ESPAsyncWebServer.h module - The web server will allow the user to control multiple devices connected with the ESP32 GPIO pins simultaneously by using HTTP GET requests to transfer the data from the user to the client. Once the HTTP URL message is sent, this is the library that helps to retrieve and process that information.

Inside the void setup() function, the following is given.

```
const char* ssid = "esp32-wifi@Knightmares";
const char* password = "12345678";
WiFi.softAP(ssid, password);
```

This creates a wifi hotspot on esp32 with the username as esp32-wifi@Knightmares and password for connecting to this as 12345678.

After this, the server.on() function can be added, which uses the Asynchronous Web Server Module to get messages depending on what is the command given in the URL.

```
server.on(<command>, HTTP_GET, [](AsyncWebServerRequest *request){
```

```
// the following code runs whenever we receive a "turn-on" request
from the webpage

    <operations to be performed when this command is encountered>
    request->send_P(200, "text/plain", "");
  });
```

This way, we can define function for moving forward “DFORWARD()” and correspondingly for the different directions and when such a command is encountered, we will call this function. An extra function “STOP()” has also been defined to not do anything. This is called when command ‘S’ is encountered.

For slider inputs, the input 1-10 is used to evaluate the spd mapping function as below, which is then used to control the speed of the bot.

```
spd = map(slidervalue, 1, 10, 100, 250);
```

After the above setup, the below line is added in the setup function to begin the server.

```
server.begin();
```

ARDUINO IDE CODE:

The functions that need to be performed are moving forward, backward, turning left and right, with a speed defined by us. We have 4 motors in our bot with the left 2 and right 2 connected to the same output of the motor driver.

When going forward, all 4 motors need to rotate in forward direction. Going backwards, all should rotate in same direction to make it move back. To make it go left, left motors must go back and right motors should go front. This is the concept of differential drive.

Functions DFORWARD(), DBACKWARD(), DLEFT(), DRIGHT() are written to implement this.

Each of the above functions send a particular command to each of the motors given by another set of functions that use Arduino command digitalWrite() to provide the required voltage to the motors through the motor driver.

For each message that we get through the server wifi connection through HTTP GET, we need to implement the corresponding above functions that provide specific voltages to the external circuit to control the required motion of the bot.