Nama: Nauval M. Fachrezi W. K.
NIM: 1103192203

```
#include <webots/motor.h>

#include <webots/robot.h>


#include <math.h>

#include <stdio.h>

#include <stdlib.h>


#define TIME_STEP 16
```

The first part of the code includes some necessary header files that are required to interface with the Webots simulation environment. The **`webots/device.h`** header file contains the definitions for various Webots devices, **`webots/motor.h`** contains the definitions for the motors, and **`webots/robot.h`** contains the main definitions for the robot. Additionally, the code also includes the **`math.h`**, **`stdio.h`**, and **`stdlib.h`** header files for mathematical calculations and standard input/output functions. The **`#define TIME_STEP 16`** statement defines a constant **`TIME_STEP`** with a value of 16, which specifies the time between each iteration of the simulation loop.

```
void my_step() {
  if (wb_robot_step(TIME_STEP) == -1) {
    wb_robot_cleanup();
    exit(EXIT_SUCCESS);
  }
}
```

The **`my_step()`** function is defined to perform a single iteration of the simulation loop with a fixed time step of **`TIME_STEP`**. The **`wb_robot_step(TIME_STEP)`** function advances the simulation by one time step and returns **`-1`** if the simulation is finished. If the simulation is finished, the **`wb_robot_cleanup()`** function is called to release the resources used by the robot and the program exits with a status of **`EXIT_SUCCESS`**.

```
int main(int argc, char **argv) {
  wb_robot_init();
```

The **`main()`** function is the entry point of the program. It initializes the robot and sets up the simulation environment using the **`wb_robot_init()`** function.

Nama: Nauval M. Fachrezi W. K.
NIM: 1103192203

```
  /* main loop */
  while (1) {
    my_step();


    /* TODO: add your code here */
  }
}
```

The main loop of the program is implemented using a **`while`** loop that runs indefinitely. Inside the loop, the **`my_step()`** function is called to advance the simulation by one time step. The **`TODO`** comment indicates that this is where the user should add their own code to control the robot.

The rest of the program is currently empty, and it is up to the user to add their own code to control the robot's motors, sensors, and other devices using the Webots API.

```
  // list devices
  int n_devices = wb_robot_get_number_of_devices();
  int i;
  printf("Available devices:\n");
  for (i = 0; i < n_devices; i++) {
    WbDeviceTag tag = wb_robot_get_device_by_index(i);
    const char *name = wb_device_get_name(tag);
    printf(" Device #%d name = %s\n", i, name);
  }
```

This section lists all the devices available in the Webots simulation environment. The **`wb_robot_get_number_of_devices()`** function returns the number of devices in the simulation, and the **`wb_robot_get_device_by_index()`** function retrieves a device tag for a device with a given index. The **`wb_device_get_name()`** function returns the name of the device. The code iterates through all the devices in the simulation and prints their names.

```
  WbDeviceTag l_arm_shx = wb_robot_get_device("LArmShx");
  WbDeviceTag r_arm_shx = wb_robot_get_device("RArmShx");
  WbDeviceTag r_arm_elx = wb_robot_get_device("RArmElx");


  double l_arm_shx_target = -1.396;
  double r_arm_shx_target = -0.77;
```

Nama: Nauval M. Fachrezi W. K.
NIM: 1103192203

This section initializes three motor devices for the left shoulder (l_arm_shx), right shoulder (r_arm_shx), and right elbow (r_arm_elx) of a robot. The **`wb_robot_get_device()`** function retrieves a device tag for a device with a given name. The code also sets two target positions for the left and right shoulders.

```
int n_steps_to_achieve__target = 1000 / TIME_STEP;  // 1 second
for (i = 0; i < n_steps_to_achieve__target; i++) {
  double ratio = (double)i / n_steps_to_achieve__target;
  wb_motor_set_position(l_arm_shx, l_arm_shx_target * ratio);
  wb_motor_set_position(r_arm_shx, r_arm_shx_target * ratio);
  my_step();
}
```

This section gradually moves the left and right shoulders to their target positions over a period of 1 second. The code calculates the number of simulation steps needed to achieve the target position and then iterates through these steps, gradually increasing the position of the left and right shoulders using **`wb_motor_set_position()`** function. The **`my_step()`** function is called at each step to advance the simulation.

```
double initTime = wb_robot_get_time();
while (true) {
  double time = wb_robot_get_time() - initTime;
  wb_motor_set_position(r_arm_elx, 0.3 * sin(5 * time) - 0.3);
  my_step();
};
```

This section moves the right elbow in a sinusoidal pattern. The code initializes a variable **`initTime`** with the current simulation time and then enters an infinite loop. Inside the loop, the code calculates the current time since the loop started and uses this time to calculate the new position of the right elbow using a sinusoidal function. The **`wb_motor_set_position()`** function is used to set the position of the right elbow, and **`my_step()`** is called at each step to advance the simulation.

Nama: Nauval M. Fachrezi W. K.
NIM: 1103192203

```
return EXIT_FAILURE;
```

This line returns a failure status when the program exits.