



Term Project

Advanced Mechatronics

Term Members
Dong Dong Liu
Mingzhe Ye
Xuchu Xu

Abstract

Disabled people or patient face many difficulties with daily life routine. Easy things like fetching a beer from refrigerator can take much more time than normal people. Especially patient with paralysation. Our project aims to design a robot to assist these group of people for better life and achieve specific goals that normally would have paid much effort.

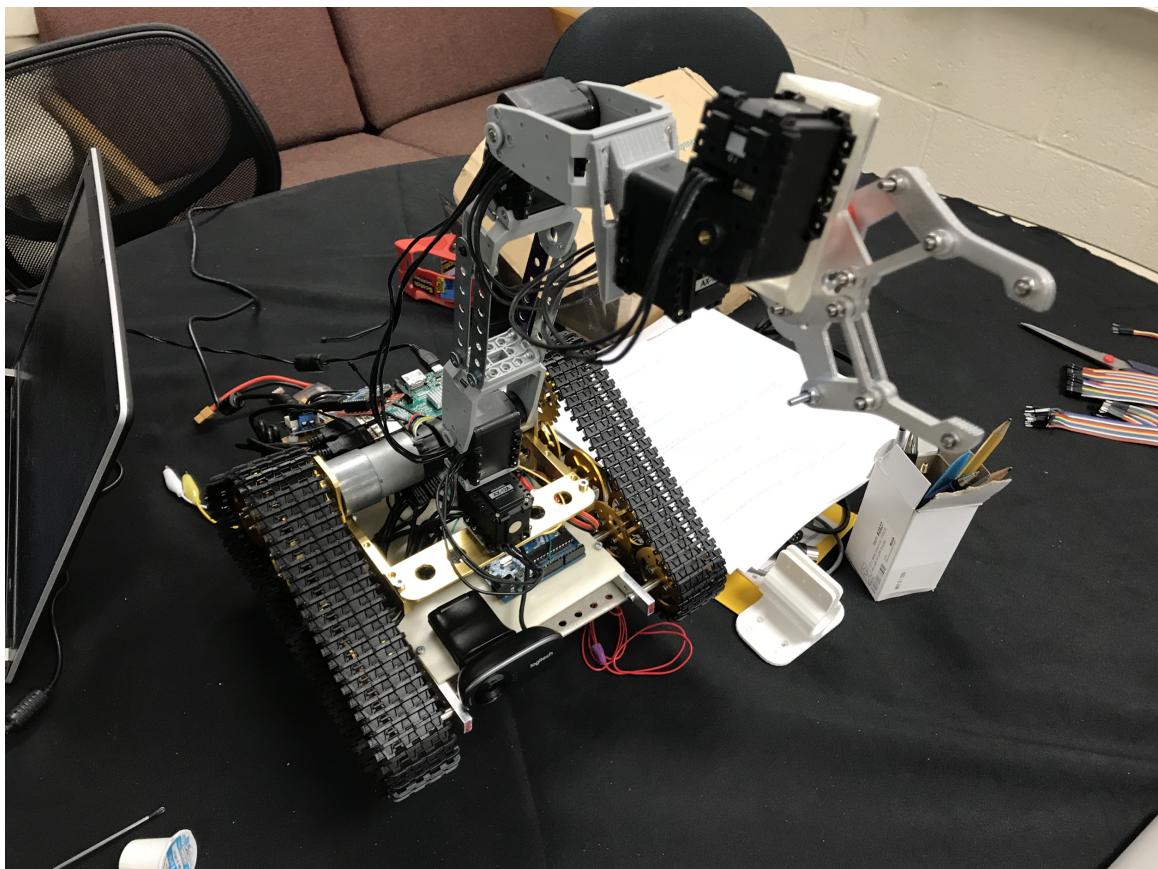
Introduction

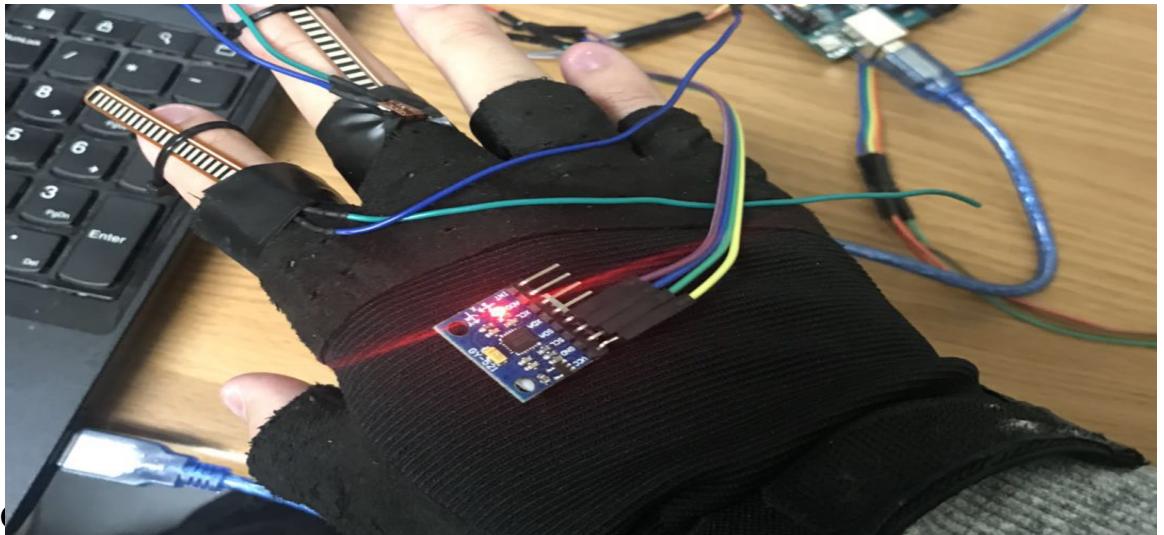
Our robot is designed to be gesture controlled with both hand. By controlling both hands with gloves, user can move the car to target destination and fetch target

object with five-joint robot arm. In the meanwhile, user can monitor robot motion by ipad with webcam in the robot chassis.

Working process

At the starting point we built up the car chassis which is caterpillar-driven car chassis. Then we developed the robot arm with five servo motors and frames. Next we did some research in studying ROS environment for controlling the servo motor. And next we set up two Xbee module configurations for wireless connection between one Arduino and raspberry Pi. Then we build up the circuits of two gyroscopes and two flex sensors with this Arduino. After that we designed the algorithm for catching the valid and stable data of IMU due to unstable data value. Next we setup webcam by using several sudo commands. Finally we connect another Arduino with Raspberry Pi in serial port connection way. This Arduino is built in car chassis for control two DC motors. With that, one H-bridge is applied. Below shows our mechanism:





Technology

As we are using two IMUs and two flex sensor, we have enough data to control our 7 DOF robot:

flex1 Elbow joint

flex2 claw joint

theta x left hand for car forward & car backward

theta y left hand for car turn left & car turn right

theta z right hand for waist joint

theta x right hand for shoulder joint

theta y right hand for wrist joint

Xbee data transformation and receiving:

```
String toTransfer = identifier_s;
for (int i =0; i < SENSOR_NUM; i++)
{
    toTransfer = toTransfer + sensor_str[i]+identifier;
}
toTransfer = toTransfer + identifier_e;
Serial.print(toTransfer);
```

ROS Working flow chart:

There are two parts about the structure in the robot except the wearing device called remote controller. The upper level called master controller is Raspberry Pi and the lower level called slave controller is Arduino. In the master controller, ROS system is running to maintain the control of robot arm and car movement, as well as being in charge of receiving the data from the remote controller via Xbee module.

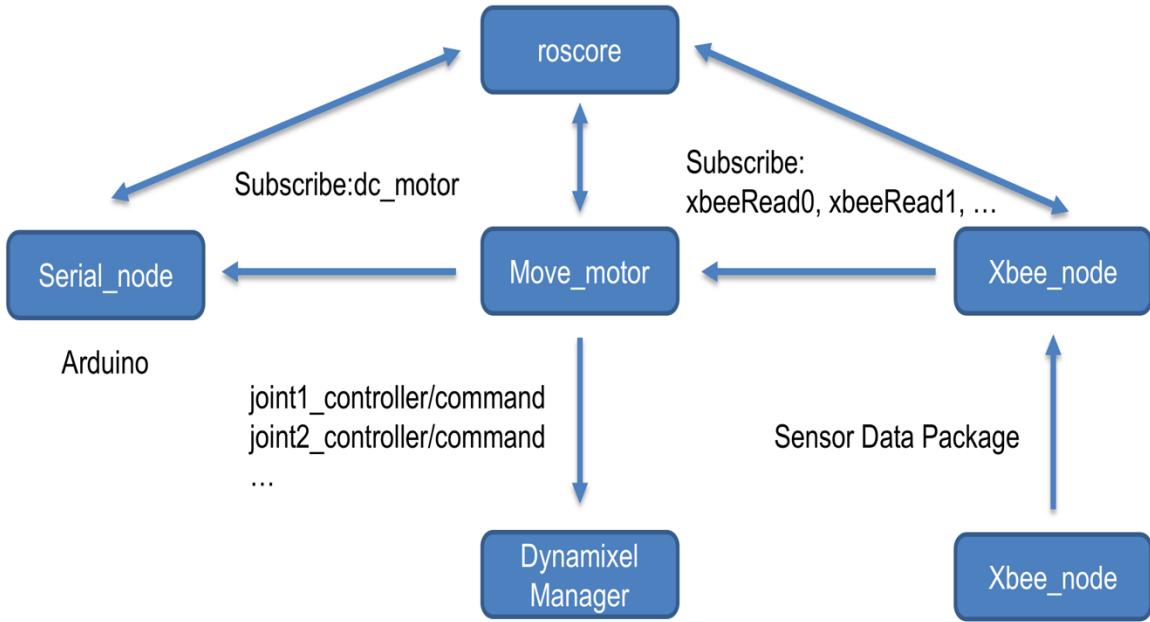
In the initial phase, the remote controller collects data from the gyro and process the data such as integration through time. Then the remote controller sends data via Xbee and the master controller gets data with serial_node maintained by roscore from another Xbee connected to master controller via USB port.

The serial_node decodes the data by identifying the delimiters contained in the transmitting package and publishes the data through topic called xbeeRead0, xbeeRead1, xbeeRead2, etc.

Table1xbee_node Topic List Definition

Topic Name	Data Type	Topic Definition
xbeeRead0	UInt32	Elbow control
xbeeRead1	UInt32	Claw control
xbeeRead2	UInt32	Car forward & backward control
xbeeRead3	UInt32	Car turn left & right
xbeeRead4	UInt32	Waist control
xbeeRead5	UInt32	Shoulder control
xbeeRead6	UInt32	Wrist control

By subscribing the topic from serial_node, move_motor node first process the raw data from Xbee, such as extreme value limit, data monitor, adjusting the controlling angle by comparing the sensor data and the actual position of motor. The node then publishes the control commands via topic dc_motor to slave controller and joint1_controller/command, joint2_controller/commands, etc., to the slave controller.



IMU algorithm and calculation:

```

//Accelerometer angle calculations
acc_total_vector = sqrt((acc_x*acc_x)+(acc_y*acc_y)+(acc_z*acc_z)); //Calculate
the total accelerometer vector
//57.296 = 1 / (3.142 / 180) The Arduino asin function is in radians
angle_pitch_acc = asin((float)acc_y/acc_total_vector)* 57.296; //Calculate the
pitch angle
angle_roll_acc = asin((float)acc_x/acc_total_vector)* -57.296; //Calculate the
roll angle
angle_yaw_acc = asin((float)acc_z/acc_total_vector)* 57.296;
//To dampen the pitch and roll angles a complementary filter is used
angle_pitch_output = angle_pitch_output * 0.9 + angle_pitch * 0.1; //Take 90% of
the output pitch value and add 10% of the raw pitch value
angle_roll_output = angle_roll_output * 0.9 + angle_roll * 0.1; //Take 90% of the
output roll value and add 10% of the raw roll value
angle_yaw_output = angle_yaw_output * 0.9 + angle_yaw * 0.1;

```

Conclusion

In this project we have got a deep understanding of ROS. With the realization of communication between Rpi and Arduino in serial port way and wireless way, we have learned a lot and experienced the efficiency of ROS and Rpi.

Appendix

See all the code files in Term project file.