THE QUAD POD
THE TRANSFORMABLE VEHICLE

Team 6
Kee Woong Haan
Jiwon Park
Zenon Son

Department of Electrical and Computer Engineering

TA: Rajarshi Roy
The Outline

- Introduction
- Block Diagram
- Mechanical and General Designs
- Electrical Parts
- The codes for the Quadpod
- Challenges and Future Works
Introduction

- The transformable vehicle with two wheels and four legs
- The vehicle that can detect obstacles to transformed into the Quadpod mode
- The vehicle fully controlled by a remote controller
- The miniature model for the new type of vehicle for unpaved, bumpy roads
Block Diagram

- IR receiver
- Sensors
- Main Station
- Power
- Remote Controller
- Power Supply (9V)
- Power Supply (5V)
- Infrared data transmission
- Pulse
- Motors
Main Frame

6 X 8 inch frame size

Top Side
- 4 Standard servo
- 2 Continuous servo
- Proximity Sensor
- 9V battery
- Arduino Mega with PCB

Bottom Side
- 2 Blue Lipo 7.4V
- Sphere Shape Wheel

Horizontal Servo
- Continuous Servo
- Wheel
- Arduino Mega with PCB on top
- 9V Battery for Arduino
- Two Blue Lipo 7.4V
- Proximity Sensor
- Sphere Wheel
Leg Frame

- 2 Standard Servo
- Angle Converter

Leg Part
Add Up
Blue Lipo Battery

Voltage: 7.4V
Max Capacity: 1.5Ah

<table>
<thead>
<tr>
<th></th>
<th>Theoretical</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>7.4V</td>
<td>7.38V</td>
</tr>
<tr>
<td>Current Capacity</td>
<td>1500mAh</td>
<td>1470mAh</td>
</tr>
<tr>
<td>Running time for our Project</td>
<td>90min</td>
<td>80min</td>
</tr>
</tbody>
</table>
9V Battery

<table>
<thead>
<tr>
<th></th>
<th>Theoretical</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>9V</td>
<td>9V</td>
</tr>
<tr>
<td>Current Capacity</td>
<td>550mAh</td>
<td>545mAh</td>
</tr>
<tr>
<td>Running time for our Project</td>
<td>5.5hours</td>
<td>5 hours</td>
</tr>
</tbody>
</table>
## Arduino Mega

<table>
<thead>
<tr>
<th>Specification</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operating Voltage</strong></td>
<td>5V</td>
</tr>
<tr>
<td><strong>Input Voltage (recommended)</strong></td>
<td>7-12V</td>
</tr>
<tr>
<td><strong>Input Voltage (limits)</strong></td>
<td>6-20V</td>
</tr>
<tr>
<td><strong>Digital I/O Pins</strong></td>
<td>54 (of which 15 provide PWM output)</td>
</tr>
<tr>
<td><strong>Analog Input Pins</strong></td>
<td>16</td>
</tr>
<tr>
<td><strong>DC Current per I/O Pin</strong></td>
<td>40 mA</td>
</tr>
<tr>
<td><strong>DC Current for 3.3V Pin</strong></td>
<td>50 mA</td>
</tr>
<tr>
<td><strong>Flash Memory</strong></td>
<td>128 KB of which 4 KB used by bootloader</td>
</tr>
<tr>
<td><strong>SRAM</strong></td>
<td>8 KB</td>
</tr>
<tr>
<td><strong>EEPROM</strong></td>
<td>4 KB</td>
</tr>
<tr>
<td><strong>Clock Speed</strong></td>
<td>16 MHz</td>
</tr>
</tbody>
</table>

*Quadpod Transform Vehicle*
DC current I/O pin

<table>
<thead>
<tr>
<th>Theoretical</th>
<th>Measured</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>40mA</td>
<td>41.288mA</td>
<td>3.12%</td>
</tr>
</tbody>
</table>

Quadpod Transform Vehicle
Servo Motor

- Operation of the Motors
  - The degree changes by pulse lengths

< Continous Servo Motor >  < Standard Servo Motor >
# PWM and Pulse Length

- **What needs to be measured**

<table>
<thead>
<tr>
<th>Pulse Length</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>750µs</td>
<td>to the left-most position</td>
</tr>
<tr>
<td>1300µs</td>
<td>clockwise rotation</td>
</tr>
<tr>
<td>1500µs</td>
<td>to the center, stop at the center</td>
</tr>
<tr>
<td>1700µs</td>
<td>counter-clockwise rotation</td>
</tr>
<tr>
<td>2250µs</td>
<td>to the right-most position</td>
</tr>
<tr>
<td>&lt; 20ms</td>
<td>to maintain a position</td>
</tr>
</tbody>
</table>

 Quadpod Transform Vehicle
PWM signals

< 750us Pulse Length >

< 1300us Pulse Length >

< 19.2ms for the gap >

< 18.7ms for the gap >
PWM signals

< 1500us Pulse Length >

< 1700us Pulse Length >

< 18.5ms for the gap >

< 18.2ms for the gap >
PWM signals

< 2250us Pulse Length >

< 17.6ms for the gap >
#include <Servo.h> // Use Servo library
Servo myServo; // Create Servo object

void setup() {
    myServo.attach(9); // Servo connected to pin 9
}

void loop() {
    myServo.writeMicroseconds(750); // 750us pulse
}

- **Issue at this stage**
  Every servo motor does not have the same default angle, so we needed to find the default angle to control the motors

  **Code for angular input:** Servo9.write(135); // rotate to 135 degree
Voltage Regulator

- To regulate voltage to 5V for servo motors (4~6V)
  - Maximum current capacity: 1.52A (8 motors × 190mA)
  - Each regulator minimum current capacity: 0.76A

- Verification Model

\[
\frac{1}{R_{tot}} = \frac{4}{30} + \frac{3}{39}, \quad R_{tot} = 4.76 \, \Omega
\]

\[
I = \frac{V}{R_{tot}} = \frac{5}{4.76} = 1.05 \, A
\]
Voltage Regulator

Regulated Voltage: 4.95V

Wiring for the Verification

Measured Total Resistance: 4.8Ω

The current flowing through the Regulator: 0.83A
Proximity Sensor

- 42kHz Ultrasonic sensor
- Operates from 2.5-5.5V
- Low 2mA supply current

< Ultrasonic Range Finder - Maxbotix LV-EZ1 >
Proximity Sensor

- **Test Code**

```cpp
int sensorPin = 0; //analog pin 0

void setup(){
    Serial.begin(9600);
}

void loop(){
    int val = analogRead(sensorPin);
    Serial.printIn(val);
    delay(100);
}
```

Quadpod Transform Vehicle
Proximity Sensor

< an object at the close distance >

< an object at the further distance >
Proximity Sensor

<table>
<thead>
<tr>
<th>Distance (Inch)</th>
<th>Measured</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>14.5</td>
</tr>
<tr>
<td><strong>10</strong></td>
<td><strong>16</strong></td>
</tr>
<tr>
<td>11</td>
<td>18</td>
</tr>
<tr>
<td>12</td>
<td>18.5</td>
</tr>
<tr>
<td>13</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
</tr>
<tr>
<td>15</td>
<td>24</td>
</tr>
<tr>
<td>16</td>
<td>26</td>
</tr>
</tbody>
</table>

< A graph drawn from the table >

< A table for the distance measurement >

Quadpod Transform Vehicle
IR receiver and Remote Controller

< IR receiver: TSOP38238 >

< Remote Controller >

< Verification Results by the given code >

Quadpod Transform Vehicle
PCB

Proximity Sensor

2 Voltage Regulator

IR Receiver
Code

- Contain two major codes
  - Vehicle mode
  - Quadpod mode
- Int “tr” was used to determine which code will be activated.
- Command “key = getIRKey()” is used to obtain signal from IR receiver.
- Depends on which button is pressed from IR remote controller, different value of key will be provided.
case 145: //("CH up") button is pressed from IR remote;
    Servo2.writeMicroseconds(1300); // Right wheel motor rotate Clockwise
    Servo1.writeMicroseconds(1700); // Left wheel motor rotate Counterclockwise
    while(val > 15) //do while distance is more than 10inch
    {
        key = getIRKey(); //Fetch the key from IR receiver
        if(key != 145) //If key is not 145, break from the loop
            break;
    }
    else{
        val = analogRead(sensorPin); //read signal from bumper button
        if(val < 16) //If the distance is less then 10inch,
            (Servo2.writeMicroseconds(1505); //Wheel motor should stop moving
            Servo1.writeMicroseconds(1515); //quadpod mode
            Servo6.write(90); // when quadpod mode start
            Servo7.write(90); //all horizontal movement motors
            Servo9.write(85); //are at 90 degree
            Servo10.write(90);
            delay(700);
            Servo2.write(165);
            Servo3.write(0); //rotate legs to the lowest position
            Servo4.write(5);
            Servo8.write(180);
            delay(700); //give 1 second delay so the vehicle will transform
            tr = 1; //tr = 1 so the code for vehicle won't be active
        }
    }
    break;
Codes for Quadpod mode

```c
//Serial.print("CH up");
Servo3.write(25); //lift front left leg 30 deg (25 deg)
Servo9.write(45); //rotate the front left leg forward (45 deg)
delay(100); //give 0.1 second delay
Servo6.write(132); //rotate rear left leg 15 deg backward (132 deg)
Servo10.write(90); //rotate front right leg 15 deg backward (90 deg)
Servo7.write(83); //rotate rear right leg 15 deg backward (83 deg)
delay(100); //give 0.1 second delay
Servo3.write(0);
delay(100);

Servo8.write(155); //lift rear left leg 30 deg (30 deg)
Servo6.write(87); //rotate rear left leg forward (135 deg)
delay(100); //give 0.1 second delay
Servo9.write(60); //rotate front left leg 30 deg backward (105 deg)
Servo7.write(68); //rotate rear right leg 30 deg backward (105 deg)
Servo10.write(75); //rotate front right leg 30 deg backward (75 deg)
delay(100); //give 0.1 second delay
Servo8.write(180);
delay(100);

Servo2.write(140); //lift Front Right leg 30 deg (150 deg)
Servo10.write(120); //rotate the front right leg forward (45 deg)
delay(100); //give 0.1 second delay
Servo9.write(75); //rotate front left leg 30 deg backward (75 deg)
Servo7.write(53); //rotate rear right leg 30 deg backward (75 deg)
Servo6.write(102); //rotate rear left leg 30 deg backward (45 deg)
delay(100); //give 0.1 second delay
Servo2.write(165);
delay(100);
```
Challenge

- Obtaining light and durable main body frame
- Keeping the balance of Quadpod while it is moving
- Weight issues
Demonstration Video

Quadpod Transform Vehicle
Recommendation

- Better Mechanical Design
- Use Stronger Servo Motor
- Multiple Proximity sensor to cover blind spot
Pulse signal needs to be provided to hold the legs up.
Using multiple proximity sensor

Current Design (1 proximity sensor)  Recommendation (2 or more sensor)
Questions?

Quadpod Transform Vehicle