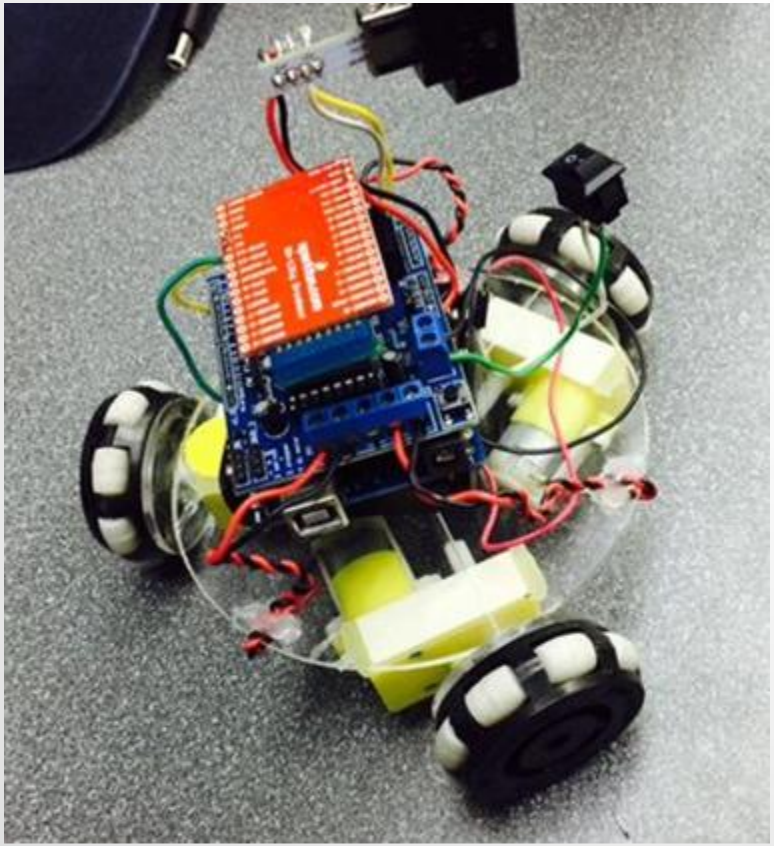
Advanced Mechatronics: Arduino Project

Control of an Arduino-Based Omni-Directional Robot Utilizing a Wii Remote and Apple iOS SDK

Anthony Brill, Matthew Moorhead, Jonghyun Bae

Omni-Directional Robot Intro



- Wirelessly controlled Omni-directional Robot
- Control Platform
  - Wii Controller (I2C Comm)
  - iPhone (TTL Serial Comm)
- Hardware Used
  - Arduino Uno
  - Wifly breakout board
  - Wii controller/receiver
  - DK Electonics Motor-shield
  - 3 Cat-Trak transwheels

Omni-Directional Robot

Theory of Operation

$$v = v_a + v_b + v_c \qquad (1)$$

$$v_a = -\omega \bar{a}_3 \times r\bar{a}_2 = \omega_A r\bar{a}_1 = -\omega_A r\bar{x} = -v_x \qquad (2)$$

$$v_b = \omega_B \bar{b}_3 \times r\bar{b}_2 = -\omega_B r\bar{b}_1 = \omega_B r\left(\frac{1}{2}\bar{x} + \frac{\sqrt{3}}{2}\bar{y}\right) = \frac{1}{2}v_x + \frac{\sqrt{3}}{2}v_y \qquad (3)$$

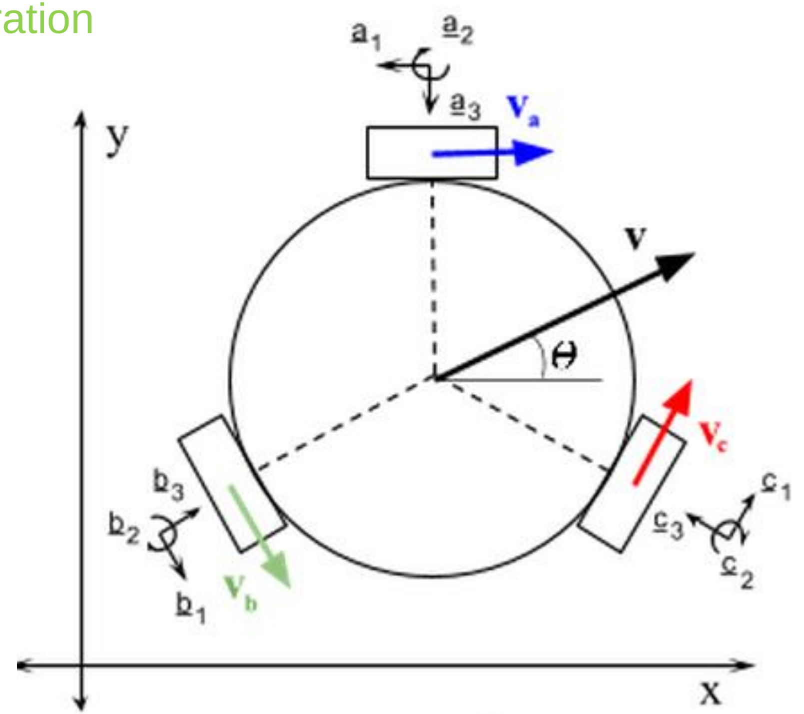
$$v_c = \omega_C \bar{c}_3 \times r\bar{c}_2 = -\omega_C r\bar{c}_1 = \omega_C r\left(\frac{1}{2}\bar{x} - \frac{\sqrt{3}}{2}\bar{y}\right) = \frac{1}{2}v_x - \frac{\sqrt{3}}{2}v_y \qquad (4)$$

$$v_x = \|v\| \cos\Theta \qquad (5)$$

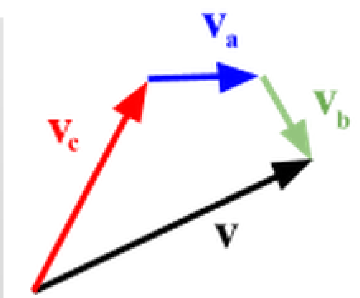
$$v_y = \|v\| \sin\Theta \qquad (6)$$

$$\Theta = \tan^{-1}\left(\frac{y}{x}\right) \qquad (7)$$

$$\|v\| = \sqrt{x^2 + y^2} \qquad (8)$$



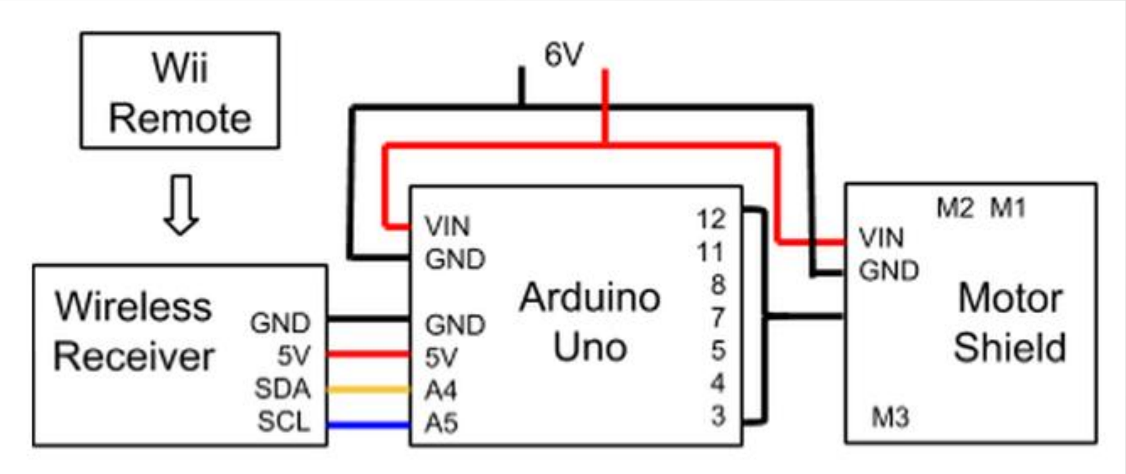
	a_1	a_2	a_3	b_1	b_2	b_3	c_1	c_2	c_3
x	-1	0	0	$\frac{1}{2}$	0	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	0	$-\frac{\sqrt{3}}{2}$
y	0	0	-1	$-\frac{\sqrt{3}}{2}$	0	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	0	$\frac{1}{2}$
z	0	-1	0	0	-1	0	0	-1	0



Omni-Directional Robot

Wii Remote

- Joystick provides x and y coordinates
- Wireless remote to receiver
- Inter-integrated Circuit (I2C) Protocol
  - Serial Clock (SCL)
  - Serial Data (SDA)



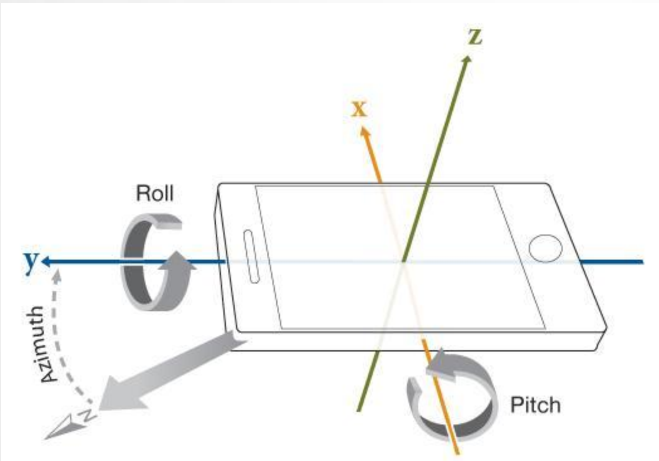
Omni-Directional Robot

Arduino_Wii Remote

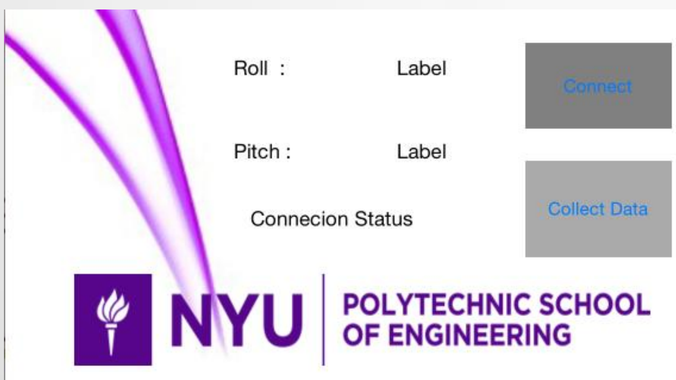
```
void loop() {

  delay(20);
  chuck.update();         // reads wii controller values

    float x = chuck.readJoyX();        // set x-position
    float y = chuck.readJoyY();        // set y-position

    float theta = atan2(y, x);         // calculate angle of direction vector
    float mag = sqrt((x*x) + (y*y));   // calculate mag of direction vector

    float vx = mag * cos(theta);       // x-component of velocity
    float vy = mag * sin(theta);       // y-component of velocity

    float w1 = -vx;                          // set wheel 1 velocity
    float w2 = 0.5 * vx - sqrt(3)/2 * vy;    // set wheel 2 velocity
    float w3 = 0.5 * vx + sqrt(3)/2 * vy;    // set wheel 3 velocity
    w1 = constrain(w1, -150, 150);
    w2 = constrain(w2, -150, 150);
    w3 = constrain(w3, -150, 150);

    boolean w1_ccw = w1 < 0 ? true : false;               // determines direction of motor spin
    boolean w2_ccw = w2 < 0 ? true : false;
    boolean w3_ccw = w3 < 0 ? true : false;
    byte w1_speed = (byte) map(abs(w1), 0, 150, 0, 255);    // maps velocity value to pwm value
    byte w2_speed = (byte) map(abs(w2), 0, 150, 0, 255);
    byte w3_speed = (byte) map(abs(w3), 0, 150, 0, 255);
```

Omni-Directional Robot

Apple iOS



- CM DeviceMotion
  - Sensor Fusion (Accelerometer, Magnetometer, Gyroscope)
  - Provides attitude / orientation of the phone
- User friendliness Orientation
  - Pitch: x-axis rotation
  - Roll: y-axis rotation
- Connection to Arduino via Async Socket
  - String formatted: $’Roll_data’#’Pitch_data$



Omni-Directional Robot

iPhone Code

```
NSString *host = @"192.168.1.122";
UInt16 port = 2000;

- (void)viewDidLoad {
    [super viewDidLoad];

    socket = [[AsyncSocket alloc] initWithDelegate:self];
    self.motionManager = [[CMMotionManager alloc] init];
    self.motionManager.showsDeviceMovementDisplay = YES;
    self.motionManager.deviceMotionUpdateInterval = 1.0 / 5.0;
    collectStatus = FALSE;
}
```

```
-(void)updateDeviceMotion
{


    CMDeviceMotion *deviceMotion = self.motionManager.deviceMotion;

    CMAttitude *attitude = deviceMotion.attitude;

    roll = (deviceMotion.attitude.roll*180/M_PI);
    pitch = (deviceMotion.attitude.pitch*180/M_PI);

    [rollAngle setText: [NSString stringWithFormat:@"%0.2f degrees", roll]];
    [pitchAngle setText: [NSString stringWithFormat:@"%0.2f degrees", pitch]];


    [self sendData];
}
```

```
-(void)sendData {
    refString = [NSString stringWithFormat:@"$%f#%f", roll, pitch];
    refData = [refString dataUsingEncoding: NSUTF8StringEncoding];
    [socket writeData:refData withTimeout:-1 tag:1];

}
```

Omni-Directional Robot

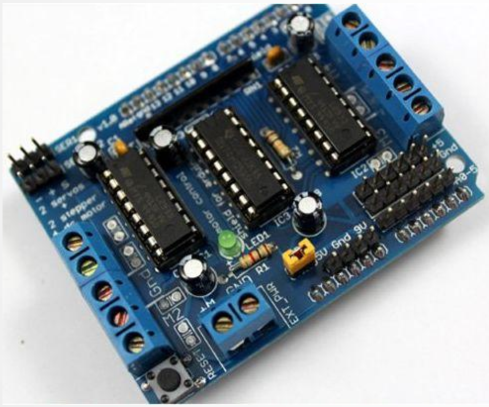
Arduino_iOS Data Parsing

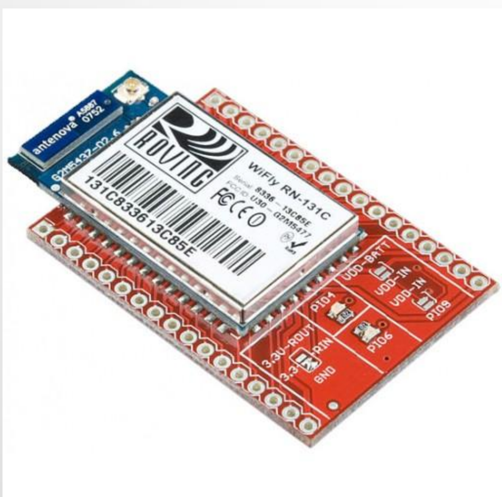
```
void ReadMsg(){
  while(mySerial.available()){
        c = mySerial.read();

        if(c == '$'){
          //if we received a $ sign then we have received a request
          incoming = 1;
        }

        while(incoming == 1){
          c = mySerial.read();
          recordMessage(c);
          delay(5);
          if ( c == '$' ) {
            incoming = 0;
```

```
              msg[letterCount] = 0;
              //moving servos here

              incomingData = atof(msg);
              incomingData2 = atof(msg2);

            }
          }
    }
    Serial.println(incomingData);
    Serial.println(incomingData2);
    letterCount = 0;
    letterCount2 = 0;
  }

  char recordMessage (char incomingMsg)
  {
    if(incomingMsg==-1 || incomingMsg=='$' ){
      state = true;
    }
    else if (incomingMsg == '#'){
      state = false;
    }
    else
   {
     if (state == true){
     msg[letterCount] = incomingMsg;
    letterCount++;
     }
     else {
       msg2[letterCount2] = incomingMsg;
       letterCount2++;
     }
   }
```

Omni-Directional Robot

Apple iOS





- Wifi communication
  - DK Electronics v1.0 motor shield requires all digital pins except 2, and 13 to operate
  - Traditional WiFly board is not compatible with this type of motor shield and an Arduino Uno
  - Software serial communication using the wifi breakout board allows pins 2 and 13 to be set as RX and TX
  - Messages are still sent over wifi to the breakout board, then transmitted to the Arduino through software serial

Omni-Directional Robot Control Platform Comparison

	User Friendliness	Intuitiveness	System Response
Wii Remote	• Wireless communication • Limited number of buttons • No motion sensing • One handed control	• No need for instruction • Familiar platform • Easy relation between direction and joystick position	• Responds quickly to real time action • Easily set to zero velocity at joystick origin • No network required
iPhone	• Wireless communication • Wireless connection displayed • Natural feeling tilt control • Ability to start and stop data transmission easily	• Small instruction set required • Background experience with motion sensing (video games) • Familiar platform • Difficult to reorient	• Drift in angle approximation • Floats around 0 degrees at iPhone reference frame • Network limitations

Conclusions

- Wii controller more stable platform
- iPhone has room for further development
  - Reduce gyroscope drift
  - Increase response time of device

- Development for the next project
  - Produce a wirelessly controlled holonomic surveillance robot controlled by an iPhone
    - Mounted camera with live video feedback
    - Slide control on the phone's screen to move the position of the camera
    - Adjustable camera position via two servo motors