F.I.N.N.
SYSTEM MANUAL

We used Python to code our robot and made use of the following library functions:

```
import time
import os
import sys
import pygame
import PicoBorgRev -> To control the motors
import usb.core
import usb.util
import cwiid       -> To control the wii remote
```

The major code blocks are as follows:

1. Establishing connection with the arm

```
while (Arm == None):
    Arm = usb.core.find(idVendor=0x1267, idProduct = 0x0000)
    Armc = Armc + 1
    if (Armc == 2200):
        print 'Could not connect to Arm, double check its connections.'
        print 'Program will continue when connection is established...'
        print ' '
        Armc = Armc/2000
        continue
```

2. Controlling the arm

```
Delay = .1
Counter = 9999
def ArmMove(Duration, ArmCmd):
    # Start the movement
    Arm.ctrl_transfer(0x40,6,0x100,0,ArmCmd,1000)

    # Stop the movement after waiting specified duration
    time.sleep(Duration)
    ArmCmd=[0,0,0]
    Arm.ctrl_transfer(0x40,6,0x100,0,ArmCmd,1000)

print 'Connected to arm succssfully!'
print ' '
print 'Press 1 and 2 on the wiimote at the same time.'
print ' sdnjdfgjodsfpogjopdsfjfgpjdsfp'
time.sleep(3)
print ''
```

3. Setting up the chassis

```
PBR = PicoBorgRev.PicoBorgRev()

PBR.Init()
if not PBR.foundChip:
    boards = PicoBorgRev.ScanForPicoBorgReverse()
    if len(boards) == 0:
        print 'No PicoBorg Reverse found, check you are attached :)'
    else:
        print 'No PicoBorg Reverse at address %02X, but we did find boards:' %
(PBR.i2cAddress)
        for board in boards:
            print '    %02X (%d)' % (board, board)
        print 'If you need to change the I²C address change the setup line so it is
correct, e.g.'
        print 'PBR.i2cAddress = 0x%02X' % (boards[0])

    sys.exit()
```

4. Ensure the communications failsafe has been enabled

```
failsafe = False
for i in range(5):
    PBR.SetCommsFailsafe(True)
    failsafe = PBR.GetCommsFailsafe()
    if failsafe:
        break
if not failsafe:
    print 'Board %02X failed to report in failsafe mode!' % (PBR.i2cAddress)
    sys.exit()
PBR.ResetEpo()
```

5. Establishing connection with the Wiimote to control the arm

```
   Wii = None
   while (Wii == None):
       print 'Establishing Connection...'
       time.sleep(1)
       try:
           Wii = cwiid.Wiimote()
           except RuntimeError:
           print 'Error connecting to the wiimote.  Press 1 and 2 on the wiimote'
           print ''

           Wii.rumble = 1
           ArmMove(.1,[0,0,1])
           ArmMove(.1,[0,0,0])
           print ''

           ArmMove(.1,[0,0,1])
           ArmMove(.1,[0,0,0])
           print 'Connection Established!'
```

```
        ArmMove(.1,[0,0,1])
        ArmMove(.1,[0,0,0])
        print ''

        ArmMove(.1,[0,0,1])
        ArmMove(.1,[0,0,0])
        print 'Press any button to continue...'

        ArmMove(.1,[0,0,1])
        ArmMove(.1,[0,0,0])
        print ''


        time.sleep(1)
        Wii.rumble = 0
        Wii.led = 15
```

6. Establishing connection with the Nunchuk for controlling the chassis and calling functions to move the arm simultaneously

```
Wii.rpt_mode = cwiid.RPT_BTN | cwiid.RPT_ACC | cwiid.RPT_EXT
Wii.state
while True:
    Accx = (Wii.state['acc'][cwiid.X])
    Accy = (Wii.state['acc'][cwiid.Y])
    Accz = (Wii.state['acc'][cwiid.Z])

    NunchukStickX=(Wii.state['nunchuk']['stick'][cwiid.X])

    NunchukStickY=(Wii.state['nunchuk']['stick'][cwiid.Y])

    NAccx = Wii.state['nunchuk']['acc'][cwiid.X]

    NAccy = Wii.state['nunchuk']['acc'][cwiid.Y]

    NAccz = Wii.state['nunchuk']['acc'][cwiid.Z]

    ChukBtn = Wii.state['nunchuk']['buttons']

    buttons = Wii.state['buttons']
```

A) The Wii remote key presses mapped to the robotic arm

If home button is pressed, exit code

```
if (buttons & cwiid.BTN_HOME):
        ArmMove(.1,[0,0,1])
        ArmMove(.1,[0,0,1])
        ArmMove(.1,[0,0,1])
        print ''
        print 'Closing Connection...'
        Wii.rumble = 1
        time.sleep(5)
        Wii.rumble = 0
        Wii.led = 0
        exit(Wii)
```

If A is pressed, close arm's grip

```
if (buttons & cwiid.BTN_A):
        print 'A pressed'
        time.sleep(Delay)
        ArmMove(.1,[1,0,0]) # Grip Close
```

If B is pressed, open arm's grip

```
if (buttons & cwiid.BTN_B):
        print 'A pressed'
        time.sleep(Delay)
        ArmMove(.1,[2,0,0]) # Grip Open
```

If button 1 is pressed, move elbow up

```
if (buttons & cwiid.BTN_1):
        print '1 pressed'
        ArmMove(.1,[16,0,0]) # Elbow Up
```

If button 2 is pressed, move elbow down

```
 if (buttons & cwiid.BTN_2):
        print '2 pressed'
        ArmMove(.1,[32,0,0]) # Elbow Down
```

If button + is pressed, move wrist up

```
if (buttons & cwiid.BTN_PLUS):
        print '+ pressed'
        ArmMove(.1,[4,0,0]) # Wrist Up
```

If button – is pressed, move wrist down

```
if (buttons & cwiid.BTN_MINUS):
        print '- pressed'
        ArmMove(.1,[8,0,0]) # Wrist Down
```

If button UP is pressed, move shoulder up

```
if (buttons & cwiid.BTN_UP):
        print 'Up Pressed'
        ArmMove(.1,[64,0,0]) # Shoulder Up
```

If button DOWN is pressed, move shoulder down

```
 if (buttons & cwiid.BTN_DOWN):
     print 'Down Pressed'
     ArmMove(.1,[128,0,0]) # Shoulder Down
```

If button LEFT is pressed, rotate base counter clockwise

```
   if (buttons & cwiid.BTN_LEFT):
      print 'Left Pressed'
         ArmMove(.1,[0,2,0]) # Rotate Base Counter-clockwise
```

If button RIGHT is pressed, rotate base clockwise

```
 if (buttons & cwiid.BTN_RIGHT):
     print 'Right Pressed'
     ArmMove(.1,[0,1,0]) # Rotate Base Clockwise
```

B) The Nunchuk movements mapped to chassis

Move towards the left

```
   if(NunchukStickX < 60):
      PBR.SetMotors(.75);

      print 'moving left'
      if (ChukBtn==1 and NunchukStickX < 60):
          PBR.SetMotors(1)
          print 'Z Pressed'
      if (ChukBtn==2 and NunchukStickX < 60):
          PBR.SetMotors(.25)
          print 'Z Pressed'
          print 'C pressed'
```

Move towards the right

```
   if(NunchukStickX > 190):
      PBR.SetMotors(-.75);
      print 'moving right' # Move towards the right
      if (ChukBtn==1 and NunchukStickX > 190):
          PBR.SetMotors(-1)
          print 'Z Pressed'
      if (ChukBtn==2 and NunchukStickX > 190):
          PBR.SetMotors(-.25)
          print 'Z Pressed'
          print 'C pressed'
```

Move forward

```
 if(NunchukStickY < 60):
     PBR.SetMotor1(-.5)
     PBR.SetMotor2(.5)
     if (ChukBtn==1 and NunchukStickY < 60):
         PBR.SetMotor1(-1)
         PBR.SetMotor2(1)
         print 'Z Pressed'
     if (ChukBtn==2 and NunchukStickY < 60):
         PBR.SetMotor1(-.25)
```

```
        PBR.SetMotor2(.25)
        print 'Z Pressed'
        print 'C pressed'
        print 'moving up' # Move forward
```

Move backward

```
  if(NunchukStickY > 190):
     PBR.SetMotor1(.5)
     PBR.SetMotor2(-.5)
     if (ChukBtn==1 and NunchukStickY > 190):
         PBR.SetMotor1(1)
         PBR.SetMotor2(-1)
         print 'Z Pressed'
     if (ChukBtn==2 and NunchukStickY > 190):
         PBR.SetMotor1(.25)
         PBR.SetMotor2(-.25)
         print 'Z Pressed'
         print 'C pressed'
         print 'moving down' # Move backward
```