

Robotics Project

Fall 2004

1. Calibration of the Infrared sensors

The first task to be accomplished is the calibration of the sensors. Since each sensor demonstrates small deviations from the absolute scale, it has to be experimentally found what values correspond to what distances.

Using the programs show below, the following rough tables have been calculated to allow for rough adjustments.

Calibrate_IR_GP2D12.osc

```
oDio1 Y = new oDio1;
oDio1 R = new oDio1;
oDio1 G = new oDio1;
oEvent Go = new oEvent;
oEvent IncrementCenterEvent = new oEvent;
Byte Enabled;

oIRRange GP2D12 = new oIRRange;
oSerial Com = new oSerial;

Sub void Main(void)
{
    Enabled = 0;

    GP2D12.IOLine = 2;
    GP2D12.Center = 20;
    GP2D12.Operate = cvTrue;

    Y.IOLine = 6;
    Y.Direction = cvInput;
    oWire GoWire = new oWire;
    GoWire.Input.Link(Y);
    GoWire.Output.Link(Go);
    GoWire.Operate = cvTrue;

    G.IOLine = 5;
    G.Direction = cvInput;
    oWire IncrementCenterWire = new oWire;
    IncrementCenterWire.Input.Link(G);
    IncrementCenterWire.Output.Link(IncrementCenterEvent);
    IncrementCenterWire.Operate = cvTrue;

    Com.Baud = cv9600;
    Com.Operate = cvTrue;

    R.IOLine = 7;
    R.Direction = cvOutput;
    R.Value = 1;

    while(1)
    {
        Tune_IR(1);
    }
}
```

```

Sub void Tune_IR(Byte Dummy)
{
    if (Enabled == 0) return;

    while (Enabled == 1)
    {
        Com.Value = GP2D12.Value;
    }
}

Sub void Go_Code(void)
{
    Enabled = 1 - Enabled;
}

Sub void IncrementCenterEvent_Code(void)
{
    GP2D12.Center = GP2D12.Center + 1;
    Com.Value = 1;
    Com.Value = GP2D12.Center;
}

```

Calibrate_IR_GPD120.osc

```

oDiol Y = new oDiol;
oDiol R = new oDiol;
oDiol G = new oDiol;
oEvent Go = new oEvent;
oEvent IncrementCenterEvent = new oEvent;
Byte Enabled;

oIRRange GP2D120 = new oIRRange;
oSerial Com = new oSerial;

Sub void Main(void)
{
    Enabled = 0;

    GP2D120.IOLine = 1;
    GP2D120.Center = 20;
    GP2D120.Operate = cvTrue;

    Y.IOLine = 6;
    Y.Direction = cvInput;
    oWire GoWire = new oWire;
    GoWire.Input.Link(Y);
    GoWire.Output.Link(Go);
    GoWire.Operate = cvTrue;

    G.IOLine = 5;
    G.Direction = cvInput;
    oWire IncrementCenterWire = new oWire;
    IncrementCenterWire.Input.Link(G);
    IncrementCenterWire.Output.Link(IncrementCenterEvent);
    IncrementCenterWire.Operate = cvTrue;

    Com.Baud = cv9600;
    Com.Operate = cvTrue;

    R.IOLine = 7;
    R.Direction = cvOutput;
}

```

```

R.Value = 1;

while(1)
{
    Tune_IR(1);
}

Sub void Tune_IR(Byte Dummy)
{
    if (Enabled == 0) return;

    while (Enabled == 1)
    {
        Com.Value = GP2D120.Value;
    }
}

Sub void Go_Code(void)
{
    Enabled = 1 - Enabled;
}

Sub void IncrementCenterEvent_Code(void)
{
    GP2D120.Center = GP2D120.Center + 1;
    Com.Value = 1;
    Com.Value = GP2D120.Center;
}

```

GD2D12

oIRRange Value	Centimeters
30	5
23	10
25	12
20	15
23	20
31	25
48	30
64	45
78	60
85	75
-128	85

GD2D120

oIRRange Value	Centimeters
37	10
58	20
79	25
92	30
104	40
-128	50

Listing of the computer program that is used to accept the values and to display them on the screen. For clarity only the code pertaining to the logic is listed. The complete file and the rest of the project are available at the download section.

```
// Fill in DCB: 57,600 bps, 8 data bits, no parity, and 1 stop bit.

dcb.BaudRate = CBR_9600;    // set the baud rate
dcb.ByteSize = 8;          // data size, xmit, and rcv
dcb.Parity = NOPARITY;     // no parity bit
dcb.StopBits = ONESTOPBIT; // one stop bit

fSuccess = SetCommState(hCom, &dcb);

if (!fSuccess)
{
    // Handle the error.
    printf ("SetCommState failed with error %d.\n", GetLastError());
    return (3);
}

printf ("Serial port %s successfully reconfigured.\n", pcCommPort);

char inBuffer[512], outBuffer[32], outBuffer1[16], outBuffer2[16];
int nBytesToRead = 16;
DWORD nBytesRead, nBytesWritten;

while(1) {
    ReadFile(hCom, &inBuffer, nBytesToRead, &nBytesRead,
(LPOVERLAPPED)NULL);
    for ( int i = 0; i < nBytesRead; i++ )
        printf("%d, ", inBuffer[i]);

    Sleep(25);
}
```

2. Calibration of the Servos

Each Servo is controlled by value between -128 and +127. The speed, however, is not linear, so we have to use a program like the one listed below to determine the optimal speeds for the robot, when going forward, and the differences of speed of the two wheels, for smooth turns. It is very helpful if the program below is run when the robot is connected to a computer, since we can see the speed output on the screen.

```
oDiol Y = new oDiol;
oDiol R = new oDiol;
oDiol G = new oDiol;

oIRRange LeftMeasure = new oIRRange;
oIRRange RightMeasure = new oIRRange;
oServoX LeftServo = new oServoX;
oServoX RightServo = new oServoX;
oSerial com = new oSerial;

oEvent Increase = new oEvent;
```

```

oEvent Decrease = new oEvent;

Sub void Main(void)
{
    Y.IOLine = 6;
    Y.Direction = cvInput;
    oWire increaseSpeed = new oWire;
    increaseSpeed.Input.Link(Y);
    increaseSpeed.Output.Link(Increase);
    increaseSpeed.Operate = cvTrue;

    G.IOLine = 5;
    G.Direction = cvInput;
    oWire decreaseSpeed = new oWire;
    decreaseSpeed.Input.Link(G);
    decreaseSpeed.Output.Link(Decrease);
    decreaseSpeed.Operate = cvTrue;

    R.IOLine = 7;
    R.Direction = cvOutput;
    R.Value = 0;

    com.Baud = cv9600;
    com.Operate = cvTrue;

    LeftServo.IOLine = 29;
    RightServo.IOLine = 30;
    RightServo.InvertOut = cvTrue;
    RightServo.Value = -20;
    LeftServo.Value = -20;

    LeftServo.Operate = cvTrue;
    RightServo.Operate = cvTrue;

    while(1)
    {

    }
}

Sub void Increase_Code(void)
{
    LeftServo.Value = LeftServo.Value + 1;
    RightServo.Value = RightServo.Value + 1;
    com.Value = LeftServo.Value;
    com.Value = RightServo.Value;
    R.Value = R.Value - 1;
}

Sub void Decrease_Code(void)
{
    LeftServo.Value = LeftServo.Value - 1;
    RightServo.Value = RightServo.Value - 1;
    com.Value = LeftServo.Value;
    com.Value = RightServo.Value;
    R.Value = R.Value - 1;
}

```

3. Avoid Obstacles with Infrared GP2D120

This first program enables the robot to avoid obstacles detected with one of the mounted infrared sensors – the Sharp GP2D120.

This sensor is fully supported by the OOPic IDE. It can determine the proximity of objects situated between 10cm and 80cm away from the robot.

This is the first step towards the complete avoid obstacles program.

Note: In order for the program to work correctly, the sensors should be put in the exact same positions and pointed at the exact same direction as when originally programmed.

avoid_obstacles_GP2D120.osc

```
GP2D120.IOLine = 1;
GP2D120.Center = 20;
GP2D120.Operate = cvTrue;

Running = 0;

while(1)
{
    /* Check if we have an obstacle in our way */
    Avoid_Obstacle(1);
}

Sub void Avoid_Obstacle(Byte Dummy)
{

    if (Running == 0) return;

    RightServo.Operate = cvTrue;
    LeftServo.Operate = cvTrue;

    while (Running)
    {

        //condition "D"
        //it is enough to check for too close
        //right since it is common for all three situaions
        if (GP2D120.Value < TooClose_GP2D120)
        {

            //Rotate in place until RightMeasure
            //is at DistanceFromLeftWall
            //and LeftMeasure is bigger than TooClose_GP2D120
            RightServo.Value = -127;
            while (GP2D120.Value < TooClose_GP2D120 )
            {
                //Rotate in place to the right
                OOPic.Delay = 20; //in 1/100 of second
            }
            RightServo.Value = 127;

        }

    }

    RightServo.Operate = cvFalse;
    LeftServo.Operate = cvFalse;

}
```

```

Sub void Go_Code(void)
{
    Running = 1 - Running;
}

```

4. Avoid obstacles with GP2D120 and GP2D12

This program takes advantage of both sensors that the robot uses like eyes. The sensors differ in their range so they have to be initialized accordingly.

The performance of this program is important since it will show how adequate is the location and direction of each of the sensors which is key factor for the success of the avoid obstacles program.

```

oDiol Y = new oDiol;
oDiol R = new oDiol;
oDiol G = new oDiol;
oEvent Go = new oEvent;
Byte Running;
oServoX LeftServo = new oServoX;
oServoX RightServo = new oServoX;
oIRRange LeftMeasure = new oIRRange; //GD2D120
oIRRange RightMeasure = new oIRRange; //GD2D12

Const TooClose_LeftSensor = 90; //right wall 30cm
Const TooClose_RightSensor = 48; //left wall 30cm
Const DistanceFromLeftWall = 70; //about 50cm

Sub void Main(void)
{
    Y.IOLine = 6;
    Y.Direction = cvInput;
    oWire GoWire = new oWire;
    GoWire.Input.Link(Y);
    GoWire.Output.Link(Go);
    GoWire.Operate = cvTrue;

    R.IOLine = 7;
    R.Direction = cvOutput;
    R.Value = 0;

    G.IOLine = 5;
    G.Direction = cvOutput;
    G.Value = 0;

    LeftServo.IOLine = 29;
    RightServo.IOLine = 30;
    RightServo.InvertOut = cvTrue;
    RightServo.Value = 127;
    LeftServo.Value = 127;

    LeftMeasure.IOLine = 1;
    LeftMeasure.Center = 20;
    LeftMeasure.Operate = cvTrue;

    RightMeasure.IOLine = 2;
    RightMeasure.Center = 20;
    RightMeasure.Operate = cvTrue;

    Running = 0;
}

```

```

while(1)
{
    /* Check if we have an obstacle in our way */
    Avoid_Obstacle(1);
}
}

Sub void Avoid_Obstacle(Byte Dummy)
{

    if (Running == 0) return;

    RightServo.Operate = cvTrue;
    LeftServo.Operate = cvTrue;

    while (Running)
    {

        //conditions "D","E", "F";
        //it is enough to check for too close
        //right since it is common for all three situaions
        if (LeftMeasure < TooClose_LeftSensor)
        {

            //Rotate in place until RightMeasure
            //is at DistanceFromLeftWall
            //and LeftMeasure is bigger than TooClose_LeftSensor
            R.Value = 1;
            RightServo.Value = -127;
            while ((RightMeasure.Value < DistanceFromLeftWall) &
(LeftMeasure.Value < TooClose_LeftSensor))
            {
                //Rotate in place to the right
                OOPic.Delay = 20; //in 1/100 of second
            }
            RightServo.Value = 127;
            R.Value = 0;

        }

        //using "ELSE" - each cycle we do only one operation for
smoothness
        else if (RightMeasure.Value < TooClose_RightSensor)
        {

            //go slightly right
            G.Value = 1;
            RightServo.Value = -127;
            while(RightMeasure < DistanceFromLeftWall)
            {
                OOPic.Delay = 10;
            }
            RightServo.Value = 127;
            G.Value = 0;

        }

    }

    RightServo.Operate = cvFalse;
    LeftServo.Operate = cvFalse;
}

Sub void Go_Code(void)
{
    Running = 1 - Running;
}

```

5. Avoid Obstacles – Complete

This is the final version of Avoid Obstacles. An improvement of the programs presented above that allows for demonstration of some independence. Here the robot uses the two sensors to let the robot navigate freely without hitting obstacles, as allowed by its sight.

avoid_obstacles.osc

```
oDiol Y = new oDiol;
oDiol R = new oDiol;
oEvent Go = new oEvent;
Byte Running;
oServoX LeftServo = new oServoX;
oServoX RightServo = new oServoX;
oIRRange LeftMeasure = new oIRRange;
oIRRange RightMeasure = new oIRRange;
Const TooCloseLeft = 48; //50cm
Const TooCloseRight = 92; //40cm

Sub void Main(void)
{
    //Initialize the button and wire
    Y.IOLine = 6;
    Y.Direction = cvInput;
    oWire GoWire = new oWire;
    GoWire.Input.Link(Y);
    GoWire.Output.Link(Go);
    GoWire.Operate = cvTrue;

    R.IOLine = 7;
    R.Direction = cvOutput;
    R.Value = 1;

    Running = 0;

    //Initialize left & right servos
    LeftServo.IOLine = 29;
    //LeftServo.InvertOut = cvTrue;
    LeftServo.Value = 127;

    RightServo.IOLine = 30;
    RightServo.InvertOut = cvTrue;
    RightServo.Value = 127;

    //Initialize left & right IR sensors
    LeftMeasure.IOLine = 1;
    LeftMeasure.Center = 20;
    LeftMeasure.Operate = cvTrue;

    RightMeasure.IOLine = 2;
    RightMeasure.Center = 20;
    RightMeasure.Operate = cvTrue;

    while(1)
    { //constantly check if we are in the moving state
      if (Running)
      {
          //Start the servos
```

```

        LeftServo.Operate = cvTrue;
        RightServo.Operate = cvTrue;

while (Running)
{
if (LeftMeasure.Value < TooCloseRight)
{
        LeftServo.Value = -127;
while (LeftMeasure.Value < TooCloseRight)
{
        }
        OOPic.Delay = 50;
        LeftServo.Value = 127;
}

else if (RightMeasure.Value < TooCloseLeft)
{
        RightServo.Value = -127;
while (RightMeasure.Value < TooCloseLeft)
{
        }
        OOPic.Delay = 50;
        RightServo.Value = 127;
}

}

} //end of while loop

RightServo.Operate = cvFalse;
LeftServo.Operate = cvFalse;

} //end if
} //end while(1)

} //end of Main() function

//control moving/not moving
Sub void Go_Code(void)
{
        Running = 1 - Running;
}

```

6. Move along the wall and avoid obstacles

Besides avoiding obstacles, through OOPic we can make the robot move following a certain pattern. Moving along a wall is a task that is applied often in practical tasks and theoretical problems, like the famous maze problem.

The solution presented includes a new task besides making sure that we do not hit any objects – to monitor a constant distance to the “wall” or set of objects which the robot is moving along. Below the program listing is a set of diagrams that describe the scheme of the different cases that are used as basis for the solution.

keep_going_along_left_wall_and_avoid_obstacles.osc

```

oDio1 Y = new oDio1;
oDio1 R = new oDio1;

```

```

oDiol G = new oDiol;
oEvent Go = new oEvent;
Byte Running;
oServoX LeftServo = new oServoX;
oServoX RightServo = new oServoX;
oIRRange LeftMeasure = new oIRRange;
oIRRange RightMeasure = new oIRRange;

Const TooClose_LeftSensor = 90; //right wall 30cm
Const TooClose_RightSensor = 48; //left wall 30cm
Const DistanceFromLeftWall = 70; //about 50cm

Sub void Main(void)
{
    Y.IOLine = 6;
    Y.Direction = cvInput;
    oWire GoWire = new oWire;
    GoWire.Input.Link(Y);
    GoWire.Output.Link(Go);
    GoWire.Operate = cvTrue;

    R.IOLine = 7;
    R.Direction = cvOutput;
    R.Value = 0;

    G.IOLine = 5;
    G.Direction = cvOutput;
    G.Value = 0;

    LeftServo.IOLine = 29;
    RightServo.IOLine = 30;
    RightServo.InvertOut = cvTrue;
    RightServo.Value = 127;
    LeftServo.Value = 127;

    LeftMeasure.IOLine = 1;
    LeftMeasure.Center = 20;
    LeftMeasure.Operate = cvTrue;

    RightMeasure.IOLine = 2;
    RightMeasure.Center = 20;
    RightMeasure.Operate = cvTrue;

    Running = 0;

    while(1)
    {
        /* Check if we have an obstacle in our way */
        Avoid_Obstacle(1);
    }
}

Sub void Avoid_Obstacle(Byte Dummy)
{
    if (Running == 0) return;

    RightServo.Operate = cvTrue;
    LeftServo.Operate = cvTrue;

    while (Running)
    {

```

```

//conditions "D","E", "F";
//it is enough to check for too close
//right since it is common for all three situaions
if (LeftMeasure < TooClose_LeftSensor)
{
    //Rotate in place until RightMeasure
    //is at DistanceFromLeftWall
    //and LeftMeasure is bigger than TooClose_LeftSensor

    RightServo.Value = -127;
    while ((RightMeasure.Value < DistanceFromLeftWall) &
(LeftMeasure.Value < TooClose_LeftSensor))
    {
        //Rotate in place to the right
        OOPic.Delay = 20; //in 1/100 of second
    }
    RightServo.Value = 127;

}
//situation "C"
//using "ELSE" - each cycle we do only one operation for
smoothness
else if (RightMeasure.Value < TooClose_RightSensor)
{
    //go slightly right

    RightServo.Value = -127;
    while(RightMeasure < DistanceFromLeftWall)
    {
        OOPic.Delay = 10;
    }
    RightServo.Value = 127;

}

//if there is no obstacle immediately forward and we are within
reasonable deviation
//from the DistanceFromLeftWall just go forward (do nothing)
else if ((RightMeasure.Value > (DistanceFromLeftWall - 10)) &
(RightMeasure.Value < (DistanceFromLeftWall + 2)))
{
    //no instructions; just keep going fwd
    OOPic.Delay = 20;
}

//in case we are we are off course to the right
else if (RightMeasure.Value > DistanceFromLeftWall - 10)
{
    //go left
    LeftServo.Value = 10;
    G.Value = 1;
    //while we approach distance and there is no obstacle to
the left
    //continuosly go to the left. This can be changed later by
going
    //in small increments if it produces smoother movement
    while((RightMeasure.Value > DistanceFromLeftWall) &
(LeftMeasure.Value > TooClose_LeftSensor))
    {
        OOPic.Delay = 10;
    }
}

```

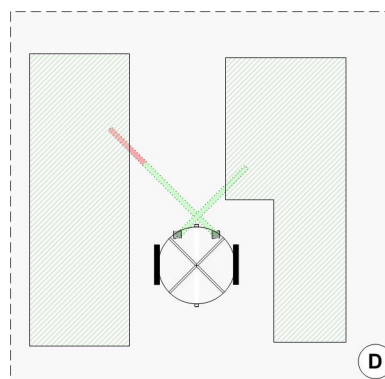
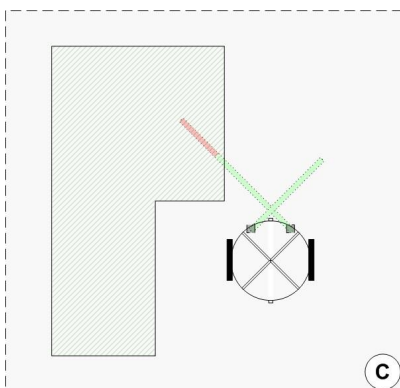
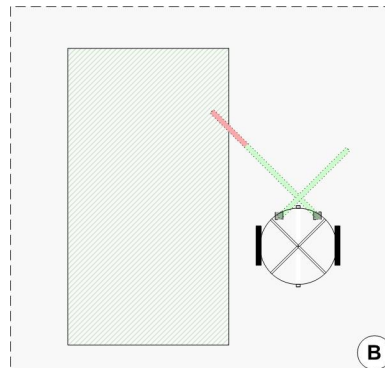
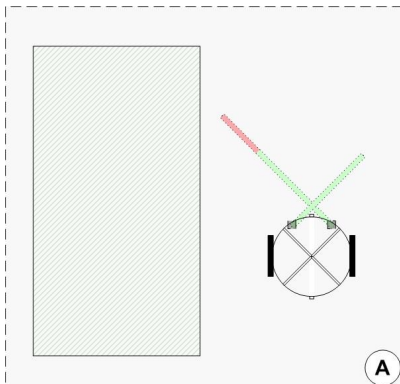
```

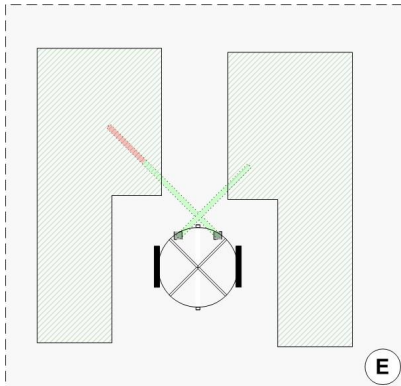
        LeftServo.Value = 127;
        G.Value = 0;
    }
    //in case we are off course to the left
    else if (RightMeasure.Value < DistanceFromLeftWall + 10)
    {
        //go to the right
        Y.Value = 1;
        RightServo.Value = -12;
        while((RightMeasure.Value > DistanceFromLeftWall) &
(LeftMeasure.Value > TooClose_LeftSensor))
        {
            OOPic.Delay = 10;
        }
        RightServo.Value = 127;
        Y.Value = 0;
    }
}

RightServo.Operate = cvFalse;
LeftServo.Operate = cvFalse;
}

Sub void Go_Code(void)
{
    Running = 1 - Running;
}

```





7. The PC Control application allows us to control directly the robot from the computer. Using the arrow keys on the keyboard we can make the robot go forward, backwards, go left, right, or turn in place.

There are actually two programs which we need: one is the host program on the robot, which receives the input from the computer, and carries out the necessary commands so that the robot goes in the specified direction. The other program is run at the computer linked to the robot. It accepts the user's input and then creates output that is suitable for the robot to read.

It is important that the information for the user's input arrives quickly. This will allow the robot to react promptly and produce smooth movement. For this purpose all the information for the user input is "packed" into one byte.

PC control.osc

```
oDio1 Y = new oDio1;
oDio1 R = new oDio1;
oEvent Go = new oEvent;
Byte Running;
oServoX LeftServo = new oServoX;
oServoX RightServo = new oServoX;
oSerial Com = new oSerial;

Sub void Main(void)
{
    Y.IOLine = 6;
    Y.Direction = cvInput;
    oWire GoWire = new oWire;
    GoWire.Input.Link(Y);
    GoWire.Output.Link(Go);
    GoWire.Operate = cvTrue;

    LeftServo.IOLine = 29;
    RightServo.IOLine = 30;
    RightServo.InvertOut = cvTrue;

    RightServo.Center = 0;
    RightServo.Offset = 0;

    LeftServo.Center = 0;
    LeftServo.Offset = 0;
}
```

```

    R.IOLine = 7;
    R.Direction = cvOutput;
    R.Value = 1;

    Running = 0;

    Com.Baud = cv9600;
    Com.Operate = cvTrue;

    while(1)
    {
        MoveRobot(1);
    }
}

Sub void MoveRobot(Byte Dummy)
{
    Byte wheel;
    Byte speed;

    if (Running == 0) {
        SetSpeed(1, 0);
        SetSpeed(2, 0);
        return;
    }

    if (Com.Received == cvTrue) {
        wheel = Com.Value;
        speed = Com.Value;
        R.Value = 0;
        SetSpeed(wheel, speed);
    }

    if (Com.Received == cvTrue) {
        wheel = Com.Value;
        speed = Com.Value;
        R.Value = 0;
        SetSpeed(wheel, speed);
    }

    SetOperation(1);
}

Sub void Go_Code(void)
{
    Running = 1 - Running;
    if (Running == 1) R.Value = 0;
    else R.Value = 1;
}

Sub void SetSpeed(Byte W, Byte V)
{
    if (W == 1) LeftServo.Value = V;
    else if (W == 2) RightServo.Value = V;
    else R.Value = 1;
}

Sub void SetOperation(Byte X)
{
    if (RightServo.Value == 0) RightServo.Operate = cvFalse;
    else RightServo.Operate = cvTrue;
    if (LeftServo.Value == 0) LeftServo.Operate = cvFalse;
    else LeftServo.Operate = cvTrue;
}

```

```
}
```

For clarity and readability, only the code pertaining to the robot interaction is included. The rest of the project is available in the download section.

```
char Direction[4] = { 0, 0, 0, 0 };
BYTE Left = 0, Right = 0;

struct WheelDrive {
    BYTE Left, Right;
};

WheelDrive WheelMatrix[] = {
    { 0, 0 }, // 0 -
    { -127, 127 }, // 1 L
    { 127, 127 }, // 2 F
    { 0, 127 }, // 3 L + F
    { 127, -127 }, // 4 R
    { 0, 0 }, // 5 R + L
    { 127, 0 }, // 6 R + F
    { 127, 127 }, // 7 R + L + F
    { -127, -127 }, // 8 B
    { 0, -127 }, // 9 L + B
    { 0, 0 }, // 10 F + B
    { -127, 127 }, // 11 F + B + L
    { -127, 0 }, // 12 B + R
    { -127, -127 }, // 13 L + R + B
    { 127, -127 }, // 14 F + B + R
    { 0, 0 }, // 15 F + B + L + R
};

int ProcessKeys(WPARAM wParam, int dir)
{
    char X[256];

    WheelDrive Drv;
    BYTE D;
    if ( (dir < 0) || (dir > 1) ) return -1;
    switch(wParam) {
        case 37:
            Direction[0] = dir;
            break;
        case 38:
            Direction[1] = dir;
            break;
        case 39:
            Direction[2] = dir;
            break;
        case 40:
            Direction[3] = dir;
            break;
    };

    D = (Direction[0] & 0x01) + ((Direction[1] & 0x01) << 1) +
        ((Direction[2] & 0x01) << 2) + ((Direction[3] & 0x01) << 3);

    Drv = WheelMatrix[D];

    Com.Send(1, Drv.Left, 2, Drv.Right);

    memset(X, 0, 256);
    itoa((signed)Drv.Left, X, 10);
}
```

```
strcpy(X + strlen(X), ", ");
itoa((signed)Drv.Right, X + strlen(X), 10);
strcpy(X + strlen(X), "          ");

HDC dc = GetDC(hWnd);

TextOut(dc, 200, 0, X, strlen(X));

ReleaseDC(hWnd, dc);

return 0;
}
```