Barak Barclay & Jasen Curtis

Kit #: 15

Leslie Tekamp

ECE 1001 - 001

30 July 2015

Gyro Bot Lab

This programs makes the bot turn 90 degrees using the gyro sensor, display the initial gyro sensor value, display the current gyro sensor value, display the degrees it needs to turn, display the degree it has turned and play a tone that changes as the gyro sensor changes. This was repeated two more times, refreshing display values. The second time it was repeated, the degree and speed was changed. The third time, the degree and speed was the same as the first time it ran through. Another task was attempted to make the bot constantly speed up until it reached a gyro sensor value and then decrease speed until it reached a gyro sensor value and loop it all. However, it didn’t work for some reason. That code is pasted as code 2.

Code:

const tSensors Gyro\_Sensor = (tSensors) S1; //gyro sensor//

task main()

{

int First\_Gyro\_Value = SensorValue[Gyro\_Sensor];

int Current\_Gyro\_Value = 0;

int Degrees = 90;

float Current\_Degree = 0;

int PlayFreq = 0;

int LowFreq = 2000;

displayTextLine(1, "%d", First\_Gyro\_Value);

motor[motorA] = 55;

motor[motorB] = -55;

while (abs(Current\_Degree) < abs(Degrees))

{

wait1Msec(10);

Current\_Gyro\_Value = SensorValue[Gyro\_Sensor] - First\_Gyro\_Value;

displayTextLine(2, "%d", Current\_Gyro\_Value);

displayTextLine(3, "%d", Degrees);

displayTextLine(4, "%d", Current\_Degree);

Current\_Degree = Current\_Degree + Current\_Gyro\_Value\*.01;

PlayFreq = LowFreq + Current\_Gyro\_Value \* 100;

playTone(PlayFreq, 20);

}

motor[motorA] = 0;

motor[motorB] = 0;

Current\_Gyro\_Value = 0;

Current\_Degree = 0;

Degrees = 720;

PlayFreq = 0;

wait1Msec(2000);

motor[motorA] = -55;

motor[motorB] = 55;

while (abs(Current\_Degree) < abs(Degrees))

{

wait1Msec(10);

Current\_Gyro\_Value = SensorValue[Gyro\_Sensor] - First\_Gyro\_Value;

displayTextLine(2, "%d", Current\_Gyro\_Value);

displayTextLine(3, "%d", Degrees);

displayTextLine(4, "%d", Current\_Degree);

Current\_Degree = Current\_Degree + Current\_Gyro\_Value\*.01;

PlayFreq = LowFreq + Current\_Gyro\_Value \* 100;

playTone(PlayFreq, 20);

}

motor[motorA] = 0;

motor[motorB] = 0;

wait1Msec(1000);

Current\_Gyro\_Value = 0;

Current\_Degree = 0;

Degrees = 90;

PlayFreq = 0;

wait1Msec(1000);

motor[motorA] = 55;

motor[motorB] = -55;

while (abs(Current\_Degree) < abs(Degrees))

{

wait1Msec(10);

Current\_Gyro\_Value = SensorValue[Gyro\_Sensor] - First\_Gyro\_Value;

displayTextLine(2, "%d", Current\_Gyro\_Value);

displayTextLine(3, "%d", Degrees);

displayTextLine(4, "%d", Current\_Degree);

Current\_Degree = Current\_Degree + Current\_Gyro\_Value\*.01;

PlayFreq = LowFreq + Current\_Gyro\_Value \* 100;

playTone(PlayFreq, 20);

}

motor[motorA] = 0;

motor[motorB] = 0;

wait1Msec(2000);

}

Code 2:

const tSensors Gyro\_Sensor = (tSensors) S1; //gyro sensor//

task main()

{

int Threshold = 0;

int Delta = 10;

int UpperThreshold =0;

int LowerThreshold =0;

int First\_Gyro\_Value = SensorValue[Gyro\_Sensor];

int Current\_Gyro\_Value = 0;

int Degrees = 90;

float Current\_Degree = 0;

int PlayFreq = 0;

int LowFreq = 2000;

displayTextLine(1, "%d", First\_Gyro\_Value);

motor[motorA] = 55;

motor[motorB] = -55;

while (abs(Current\_Degree) < abs(Degrees))

{

wait1Msec(10);

Current\_Gyro\_Value = SensorValue[Gyro\_Sensor] - First\_Gyro\_Value;

displayTextLine(2, "%d", Current\_Gyro\_Value);

displayTextLine(3, "%d", Degrees);

displayTextLine(4, "%d", Current\_Degree);

Current\_Degree = Current\_Degree + Current\_Gyro\_Value\*.01;

PlayFreq = LowFreq + Current\_Gyro\_Value \* 100;

playTone(PlayFreq, 20);

}

motor[motorA] = 0;

motor[motorB] = 0;

Current\_Gyro\_Value = 0;

Current\_Degree = 0;

Degrees = 720;

PlayFreq = 0;

wait1Msec(2000);

motor[motorA] = -100;

motor[motorB] = 100;

while (abs(Current\_Degree) < abs(Degrees))

{

wait1Msec(10);

Current\_Gyro\_Value = SensorValue[Gyro\_Sensor] - First\_Gyro\_Value;

displayTextLine(2, "%d", Current\_Gyro\_Value);

displayTextLine(3, "%d", Degrees);

displayTextLine(4, "%d", Current\_Degree);

Current\_Degree = Current\_Degree + Current\_Gyro\_Value\*.01;

PlayFreq = LowFreq + Current\_Gyro\_Value \* 100;

playTone(PlayFreq, 20);

}

motor[motorA] = 0;

motor[motorB] = 0;

wait1Msec(1000);

Current\_Gyro\_Value = 0;

Current\_Degree = 0;

Degrees = 90;

PlayFreq = 0;

wait1Msec(1000);

motor[motorA] = 55;

motor[motorB] = -55;

while (abs(Current\_Degree) < abs(Degrees))

{

wait1Msec(10);

Current\_Gyro\_Value = SensorValue[Gyro\_Sensor] - First\_Gyro\_Value;

displayTextLine(2, "%d", Current\_Gyro\_Value);

displayTextLine(3, "%d", Degrees);

displayTextLine(4, "%d", Current\_Degree);

Current\_Degree = Current\_Degree + Current\_Gyro\_Value\*.01;

PlayFreq = LowFreq + Current\_Gyro\_Value \* 100;

playTone(PlayFreq, 20);

}

motor[motorA] = 0;

motor[motorB] = 0;

wait1Msec(2000);

motor[motorA] = 20;

motor[motorB] = 20;

wait1Msec(1000);

int Current\_Gyro\_Value\_2 = SensorValue[Gyro\_Sensor];

Threshold = Current\_Gyro\_Value\_2;

UpperThreshold = Threshold + Delta;

LowerThreshold = Threshold - Delta;

int Motor\_Speed = 0;

int Motor\_Speed\_2 = 0;

int x = time100[T1];

int Current\_Motor\_Speed;

while(x < 6000)

{

Current\_Gyro\_Value\_2 = SensorValue[Gyro\_Sensor];

if (SensorValue[S3] > UpperThreshold)

{

Motor\_Speed++;

Current\_Motor\_Speed = 20 + Motor\_Speed;

motor[motorA] = Current\_Motor\_Speed;

motor[motorB] = Current\_Motor\_Speed;

wait1Msec(500);

}

if (SensorValue[S3] < LowerThreshold)

{

Motor\_Speed\_2++;

Current\_Motor\_Speed = 20 - Motor\_Speed\_2;

motor[motorB] = Current\_Motor\_Speed;

motor[motorA] = Current\_Motor\_Speed;

wait1Msec(500);

}

}

}