Arm and Gripper Programming Guide (ROBOTC®)

Introduction:
In this guide, the Ranger Bot will be programmed to follow a line, pick up an object, and put it in a box. This guide is for use with the ROBOTC® programming language.

Getting Started:
1. To start the program, type the Main Task function, followed by an opening brace.

   ```
   task main()
   {
   }
   ```

Defining Variables:
2. Declare five integer variables as seen in the code below. Beside each variable below, there are two slashes (//) followed by what that variable is representing. These mark a comment. Notice that comments appear in green when code is typed into ROBOTC.

   ```
   int nValue;      // light value
   int mValue;      // sonar value
   int aPosition;   // servo arm position
   int gPosition;   // gripper servo position
   int mSpeed;      // motor speed
   ```

3. Type the Wait function and set it to wait 50 milliseconds to allow the robot to initialize motors and sensors.

   ```
   wait1Msec(50);
   ```

4. Type the servo position command to initialize the position of both servos. This must be set to a value between 0 and 255, representing a position between 0° and 180°. This is the servo's full range of motion.

   ```
   servo[servo1] = 200;
   servo[servo2] = 100;
   ```

5. Type the Wait function and set it to wait 1 second while the servo motors get into position.

   ```
   wait1Msec(1000);
   ```

6. Add an infinite While Loop to make the containing code execute forever.

   ```
   while(true)
   {
   }
Arm and Gripper Programming Guide (ROBOTC®)

7. Set the defined variables to a value and display them on the screen.
   a. Set the nValue variable to the value of the light sensor and the mValue variable to the value of the ultrasonic sensor.
   b. Use the NXT write to screen command to put them on the screen.

   **Note:** This command requires three pieces of information (called parameters) in order to write both text and a variable to the screen:
   i. The line number
   ii. The text (in quotations)
   iii. The text (in quotations) which includes a display variable caller.

   **Note:** In this case, %d is the display variable caller. In the place of %d, an integer variable is displayed on the screen.

   This information is inside the brackets following the command, separated by commas, as seen below.

   ```plaintext
   nValue = SensorValue[LightSensor];
   nxtDisplayTextLine(0, "Light Value: %d", nValue);
   mValue = SensorValue[SonarSensor];
   nxtDisplayTextLine(2, "Sonar Value: %d", mValue);
   ```

8. Below the text now written on the screen, display a message that says **"Searching For Target"**. Because a variable will not be displayed, only two parameters will need to be sent to the function: the line number and the text.

   ```plaintext
   nxtDisplayCenteredTextLine(4, "Searching");
   nxtDisplayCenteredTextLine(5, "For Target");
   ```

9. The next few lines of code will decrease the motor speed as the target is being approached. Add a **conditional statement** so that if the ultrasonic sensor value is greater than 30, the mSpeed variable will be set to 30, but if it is equal to or less than 30, the mSpeed variable will be set to 15.

   ```plaintext
   if(SensorValue[SonarSensor] > 30)
   {
       mSpeed = 30;
   }
   else
   {
       mSpeed = 15;
   }
   ```
Arm and Gripper Programming Guide (ROBOTC®)

Line Follower Code:
10. Add the line-following code that was created in Line Follower Extension Part 3.

```c
if(SensorValue[LightSensor] < 50)
{
    motor[motorD] = mSpeed;
    motor[motorE] = 0;
}
else
{
    motor[motorE] = mSpeed;
    motor[motorD] = 0;
}
```

Grab an Object:
11. Add a **conditional statement** to check if the robot is within 17 cm of the object it is picking up. If the robot is within 17 cm of the object, display “**Found! Grabbing...**” on the robot screen and stop the motors.

```c
if(SensorValue[SonarSensor] < 17)
{
    nxtDisplayCenteredTextLine(4, "Found! Grabbing...");
    nxtDisplayCenteredTextLine(5, "Grabbing...");
    motor[motorE] = 0;
    motor[motorD] = 0;
}
```

12. Add a **Wait** function to pause for one second to give the motors time to stop before the next action is executed.

```c
wait1Msec(1000);
```

13. Add a **For Loop** which makes the arm servo move from its current position at 100 to a lower position of 190, waiting 10 milliseconds between each movement.

```c
for(aPosition = 100; aPosition < 190; aPosition++)
{
    servo[servo2] = aPosition;
    wait10Msec(1);
}
```

14. Type the **Wait** function to make the program pause for **0.5 seconds**.

```c
wait1Msec(500);
```
15. Add another For Loop to make the gripper servo move from its current position at 180 to a position of 80, waiting 10 milliseconds between each movement in order to close the gripper.

```c
for(gPosition = 180; gPosition > 80; gPosition--)
{
    servo[servo1] = gPosition;
    wait10Msec(1);
}
```

16. Type the Wait function to make the program pause for 0.5 seconds.

```c
wait1Msec(500);
```

17. Add a For Loop to make the arm servo move from its current position at 160 to a higher position of 25, waiting 10 milliseconds between each movement.

```c
for(aPosition = 160; aPosition > 25; aPosition--)
{
    servo[servo2] = aPosition;
    wait10Msec(1);
}
```

18. Add a conditional statement to check if the value of the touch sensor is 1 (or pressed). If it is, stop both motors and display the light and ultrasonic sensor values on the display. Also, display text that says “Status: Got it!”, play a sound (called “soundBlip”), and waits one second for the motors to stop completely.

```c
if(SensorValue[Touch] ==1)
{
    motor[motorE] = 0;
    motor[motorD] = 0;
    nValue = SensorValue[LightSensor];
    nxtDisplayTextLine(0, "Light Value: %d", nValue);
    mValue = SensorValue[SonarSensor];
    nxtDisplayTextLine(2, "Sonar Value: %d", mValue);
    nxtDisplayCenteredTextLine4, ("Status:");
    nxtDisplayCenteredTextLine5, ("Got It!");
    PlaySound(soundBlip);
    wait1Msec(1000);
}
```

19. Add another conditional statement to check if the value of the touch sensor is 0 (or unpressed). If it is, stop both motors and display the light and ultrasonic sensor values to the display.
20. Also, display text that says "\textit{Status: Missed It!}" and waits 0.5 seconds for the motors to stop completely.

\begin{verbatim}
if(SensorValue[Touch] ==0)
{
    motor[motorE] = 0;
    motor[motorD] = 0;
    nValue = SensorValue[LightSensor];
    nxtDisplayTextLine(0, "Light Value: \%d", nValue);
    mValue = SensorValue[SonarSensor];
    nxtDisplayTextLine(2, "Sonar Value: \%d", mValue);
    nxtDisplayCenteredTextLine(4, "Status:");
    nxtDisplayCenteredTextLine(5, "Missed It!");
    waitMsec(500);
}
\end{verbatim}

\textbf{End the Program:}

21. Type a \textbf{closing brace} to end the conditional statement in Step 11.

\begin{verbatim}
}
\end{verbatim}

22. Type a \textbf{closing brace} to end the infinite While Loop from Step 6.

\begin{verbatim}
}
\end{verbatim}

23. Type a \textbf{closing brace} to end the program.

\begin{verbatim}
}
\end{verbatim}

\textbf{Completed Code:}

\begin{verbatim}
task main()
{
    int nValue; // light value
    int mValue; // sonar value
    int aPosition; // servo arm position
    int gPosition; // gripper servo position
    int mSpeed; // Motor Speed

    waitMsec(50); // The program waits 50 milliseconds to initialize the light sensor.

    servo[servo1] = 200; // Initial startup gripper servo position
    servo[servo2] = 80; // Initial startup arm servo position
}
\end{verbatim}
mSpeed = 30;

waitMsec(1000); // wait for arm and gripper to get to position

while(true) // Infinite loop
{
    nValue = SensorValue[LightSensor];
    nxtDisplayTextLine(0, "Light Value: %d", nValue);
    mValue = SensorValue[SonarSensor];
    nxtDisplayTextLine(2, "Sonar Value: %d", mValue);
    nxtDisplayCenteredTextLine(4, "Searching");
    nxtDisplayCenteredTextLine(5, "For Target");
}

// Decrease speed as we approach target

if(SensorValue[SonarSensor] > 30)
{
    mSpeed = 30;
}
else
{
    mSpeed = 15;
}

/// Line following Here;

if(SensorValue[LightSensor] < 50) // If the Light Sensor reads a value less than 50:
{
    motor[motorD] = mSpeed; // Motor D is run at a 30 power level.
    motor[motorE] = 0; // Motor E is run at a 0 power level.
}
else // If the Light Sensor reads a value greater than or equal to 50:
{
    motor[motorE] = mSpeed; // Motor E is run at a 30 power level.
    motor[motorD] = 0; // Motor D is run at a 0 power level.
}

/// Sonar ranging here; Stop at target distance, position gripper and arm

if(SensorValue[SonarSensor] < 17) // Target sense distance (centimeters)
{  
    nxtDisplayCenteredTextLine(4, "Found!");  // update display  
    nxtDisplayCenteredTextLine(5, "Grabbing..");  // update display  
    motor[motorE] = 0;  // Motor B is off  
    motor[motorD] = 0;  // Motor C is off  
    waitMsec(1000);  // Pause for 1 second  

    for (aPosition = 100; aPosition < 190; aPosition++)  
    {  
        servo[servo2] = aPosition;  // lower arm  
        wait0Msec(1);  
    }  

    waitMsec(500);  // Pause for 1/2 second  

    for (gPosition = 180; gPosition > 80; gPosition--)  
    {  
        servo[servo1] = gPosition;  // close gripper  
        wait0Msec(1);  
    }  

    waitMsec(500);  // Pause for 1/2 second  
    for (aPosition = 160; aPosition > 25; aPosition--)  
    {  
        servo[servo2] = aPosition;  // raise arm  
        wait0Msec(1);  
    }  

    if(SensorValue[Touch] ==1)  // check gripper sensor status  
    {  
        motor[motorE] = 0;  // Motor B is off  
        motor[motorD] = 0;  // Motor C is off  

        nValue = SensorValue[LightSensor];  
        nxtDisplayTextLine(0, "Light Value: %d", nValue);  // update display  
        mValue = SensorValue[SonarSensor];  
        nxtDisplayTextLine(2, "Sonar Value: %d", mValue);  // update display  
        nxtDisplayCenteredTextLine(4, "Status: ");  // update display  
        nxtDisplayCenteredTextLine(5, "Got It!");  // update display  
        PlaySound(soundBlip);  // Play the sound, 'soundBeepBeep'.  
        waitMsec(1000);  
    }  
}
if(SensorValue[Touch] == 0) // check gripper sensor status
{
    motor[motorE] = 0; // Motor B is off
    motor[motorD] = 0; // Motor C is off
    nValue = SensorValue[LightSensor];
    nxtDisplayTextLine(0, "Light Value: %d", nValue); // update display
    mValue = SensorValue[SonarSensor];
    nxtDisplayTextLine(2, "Sonar Value: %d", mValue); // update display
    nxtDisplayCenteredTextLine(4, "Status:"); // update display
    nxtDisplayCenteredTextLine(5, "Missed It!"); // update display
    wait1Msec(500);
}

}