BashaBot Version 2.0
System Documentation Version 1.2
September 28, 2018

Caveat: As you work with the robot, please let me know of any issues, improvements, or extensions. It is a work in progress.

Hardware:

The robot operates on differential drive; each wheel turns independently, at a different speed, and has a built-in encoder sensor to measure that speed. The number of sensor ticks defines each wheel speed, resulting in some forward velocity and angular velocity over some period of time.

Additionally, the base platform supports four IR distance sensors. Currently, each robot has only three installed: left, middle, and right. Each sensor measures a distance range of 4cm – 30 cm.

One battery (either two-cell Lithium-Ion or NiMH) powers the entire system. The current batteries have a capacity of 2600 mAH for the LiIon. As the capacity reduces, the battery voltage will drop; the system will shut down immediately once the voltage reaches 6V. You can recharge the battery using the charger located in Anderson 105.

There is a red LED that acts as a heartbeat to indicate the TI microcontroller is working. If this stops flashing, reset the TI using the button on the top board. You can also use this reset if you think the board has entered a funny state. It will not reset the Raspberry Pi; you can use the power switch to hard reset the entire system if needed.

Software

You will receive the robot with the TI already programmed to control and measure the base hardware. It will communicate with the Raspberry Pi via an UART. This will require that you disconnect the Raspberry Pi UART from the console and enable the UART. We did this for you on your current image.

I have written code in Python for the Raspberry Pi that performs the communication portion, including the communication protocol. This is robotControl.py. There is exampleRobotMain.py that shows has all functions called to demonstrate how to use the different options (all but one are commented out).

For all direct move functions (XY and Rotate), the robot checks for obstacles in performing the motion. It will return a flag if it stopped due to obstacles.

The system operates with split request and reply messages. First you send a message requesting a movement, measurement, or some other behavior. Then you

read the reply message. This allows you to send a lengthy movement command while still reading the IR sensors, for example.

You need to import RobotControl and also open the serial port before calling any functions. Please also make sure to close the serial port.

API

Please carefully follow the API. Error checking is not completed for all functions. Most will still run if you provide the wrong input; it could put the system into a funny state.

moveRobotXY:

This will move the robot to a (X, Y) position relative to the robot. X and Y must be integers within (0, 65535). Currently this function only supports positive X (a negative X will work, but the robot does not rotate to face the expected direction). If you provide (X, 0) coordinates, the robot will move precisely straight with no angle. Translating the coordinates into robot movements does involve approximations and rounding so the robot may not move precisely to the position given (it will move less than expected). The function returns a flag indicating if the function stopped due to an obstacle (obsFlag = 1) or completed the requested movement (0), the distance traveled by each wheel in centimeters, and the time in milliseconds.

```
# Move robot to X,Y where X is positive
# Reply consists of a named tuple in the order (obsFlag, rightWheelDist, leftWheelDist, time)
robotControl.moveRobotXY(serialPort,  x, y)
result = robotControl.readResponse(serialPort)
time.sleep(.5)
print(result.obsFlag, result.rightWheelDist, result.leftWheelDist, result.time)
```

moveRobotObs:

This moves the robot until it encounters an obstacle. The function returns a flag indicating if the function stopped due to an obstacle (obsFlag = 1) or completed the requested movement (0), the distance traveled by each wheel in centimeters, and the time in milliseconds.

```
# Move robot until obstacle seen
# Reply consists of a named tuple in the order (obsFlag, rightWheelDist, leftWheelDist, time)
robotControl.moveRobotObs(serialPort)
result = robotControl.readResponse(serialPort)
time.sleep(.5)
print(result.obsFlag, result.rightWheelDist, result.leftWheelDist, result.time)
```

rotateRobot:

This function will rotate the robot by theta. Theta must be in radians between (0, 2pi). The function returns a flag indicating if the function stopped due to an obstacle (obsFlag = 1) or completed the requested movement (0), the distance traveled by each wheel in centimeters, and the time in milliseconds.

```
# Rotate robot by angleToTurn (0 to 2pi)
# Reply consists of a named tuple in the order (obsFlag, rightWheelDist, leftWheelDist, time)
robotControl.rotateRobot(serialPort , angleToTurn)
result = robotControl.readResponse(serialPort)
time.sleep(.5)
print(result.obsFlag, result.rightWheelDist, result.leftWheelDist, result.time)
```

getSensors:

This function reads all three IR sensors. Sensor values are returned in ADC increments.

```
# Reads the IR sensors
# Reply consists of a named tuple in the order (left, middle, right)
 robotControl.getSensors(serialPort)
result, msgType = robotControl.readResponse(serialPort)
time.sleep(.5)
print(result)
```

changePWMValues:

This function changes the left and right PWM values used by the wheels. There is no response.

```
# Modifies the PWM values used by the wheels
robotControl.ChangePWMvalues(ser, leftWheel, rightWheel)
```