Team Jbot presentation

By: Jamison Monroe
Advisor: Dr. Charles Kim

4/20/2021

Problem Statement

- Problem Statement: Throughout the course of this project, I want to establish the tasks that the Bot would fulfill and create a user interface for the Bot, potentially come up with some advanced functionality for the project, and successfully enable humans to oversee and interact with the Bot while also being able to use machine learning to advance its functions.

Design Requirements

- Product Specifications:

- Application Software Robot Operating System (ROS) software, Jetson Nano
- Memory 4 GB 64-bit LPDDR4
- Storage Micro SD card slot
- USB 4x USB 3.0, USB 2.0
- Camera 136° FOV camera for machine vision
- Power Micro USB (5V 2A), DC jack (5V 4A), 3x 18650 batteries
- Dimensions Core module: 69.6 mm × 45 mm, Whole kit: 100mm × 80mm × 29mm

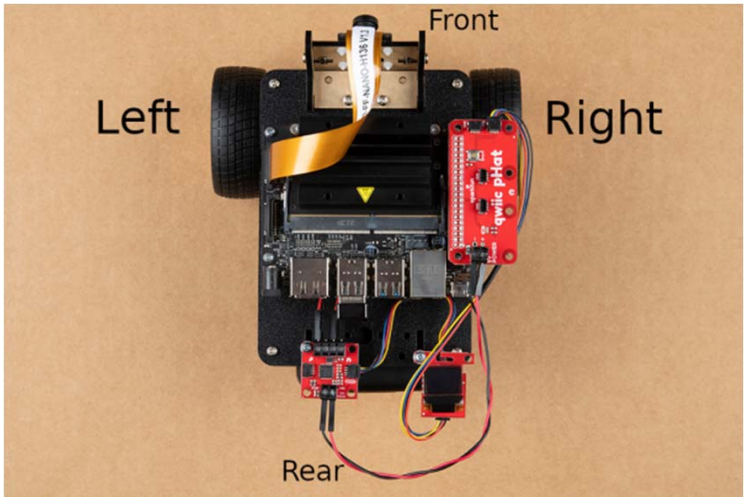
Top Solution Design

- Sleeping Assistant Bot:

- Must be able to detect sleeping activity of user as well as other objects around the house in order to navigate throughout rooms of interest, and it should be able to notify the user

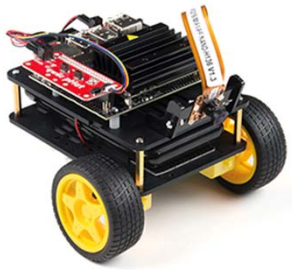
- Features:

  - Autonomous navigation
  - Object detection
  - Camera and speaker for Audio/Video transmission
  - Energy saving mode
  - Notification capability



Schematic

Sprint #1



is ordered and on the way

ot kit as well as other required parts such as camera and infrared array is ordered

- Week 2: Get shipment information and delivery time of parts

Sprint #2

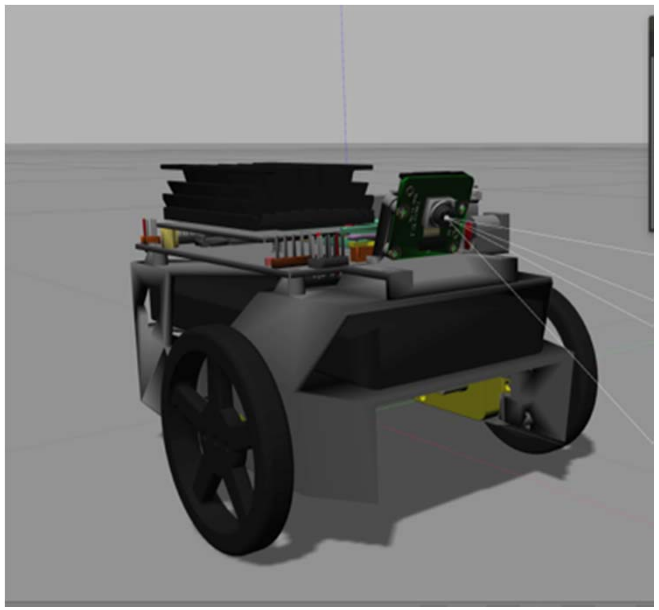
- Create python code for bot that allows it to travel around an area autonomously
- Week 1: Write and test code that operates bot and gets it moving
- Week 2: Write and test code that allows bot to move more autonomously

```
import rospy
import time

from Adafruit_MotorHAT import Adafruit_MotorHAT
from std_msgs.msg import String



# sets motor speed between [-1.0, 1.0]
def set_speed(motor_ID, value):
        max_pwm = 115.0
        speed = int(min(max(abs(value * max_pwm), 0), max_pwm))

        if motor_ID == 1:
                motor = motor_left
        elif motor_ID == 2:
                motor = motor_right
        else:
                rospy.logerror('set_speed(%d, %f) -> invalid motor_ID=%d', motor_ID, value, motor_ID)
                return

        motor.setSpeed(speed)

        if value > 0:
                motor.run(Adafruit_MotorHAT.FORWARD)
        else:
                motor.run(Adafruit_MotorHAT.BACKWARD)
```

```
# stops all motors
def all_stop():
        motor_left.setSpeed(0)
        motor_right.setSpeed(0)

        motor_left.run(Adafruit_MotorHAT.RELEASE)
        motor_right.run(Adafruit_MotorHAT.RELEASE)


# directional commands (degree, speed)
def on_cmd_dir(msg):
        rospy.loginfo(rospy.get_caller_id() + ' cmd_dir=%s', msg.data)

# raw L/R motor commands (speed, speed)
def on_cmd_raw(msg):
        rospy.loginfo(rospy.get_caller_id() + ' cmd_raw=%s', msg.data)

# simple string commands (left/right/forward/backward/stop)
def on_cmd_str(msg):
        rospy.loginfo(rospy.get_caller_id() + ' cmd_str=%s', msg.data)

        if msg.data.lower() == "left":
                set_speed(motor_left_ID,  -1.0)
                set_speed(motor_right_ID,  1.0)
        elif msg.data.lower() == "right":
                set_speed(motor_left_ID,   1.0)
                set_speed(motor_right_ID, -1.0)
        elif msg.data.lower() == "forward":
                set_speed(motor_left_ID,   1.0)
                set_speed(motor_right_ID,  1.0)
        elif msg.data.lower() == "backward":
                set_speed(motor_left_ID,  -1.0)
                set_speed(motor_right_ID, -1.0)
        elif msg.data.lower() == "stop":
                all_stop()
        else:
                rospy.logerror(rospy.get_caller_id() + ' invalid cmd_str=%s', msg.data)
```

Sprint #3

- Create python code for bot to use its camera to be able to travel from room to room as well as detect if a person is present and if they are sleeping or not
- Week 1: Write and test code that uses the camera attached to bot to help it identify objects so that it can move from room to room
- Week 2: Write and test code that uses the infrared array attachment to detect heat



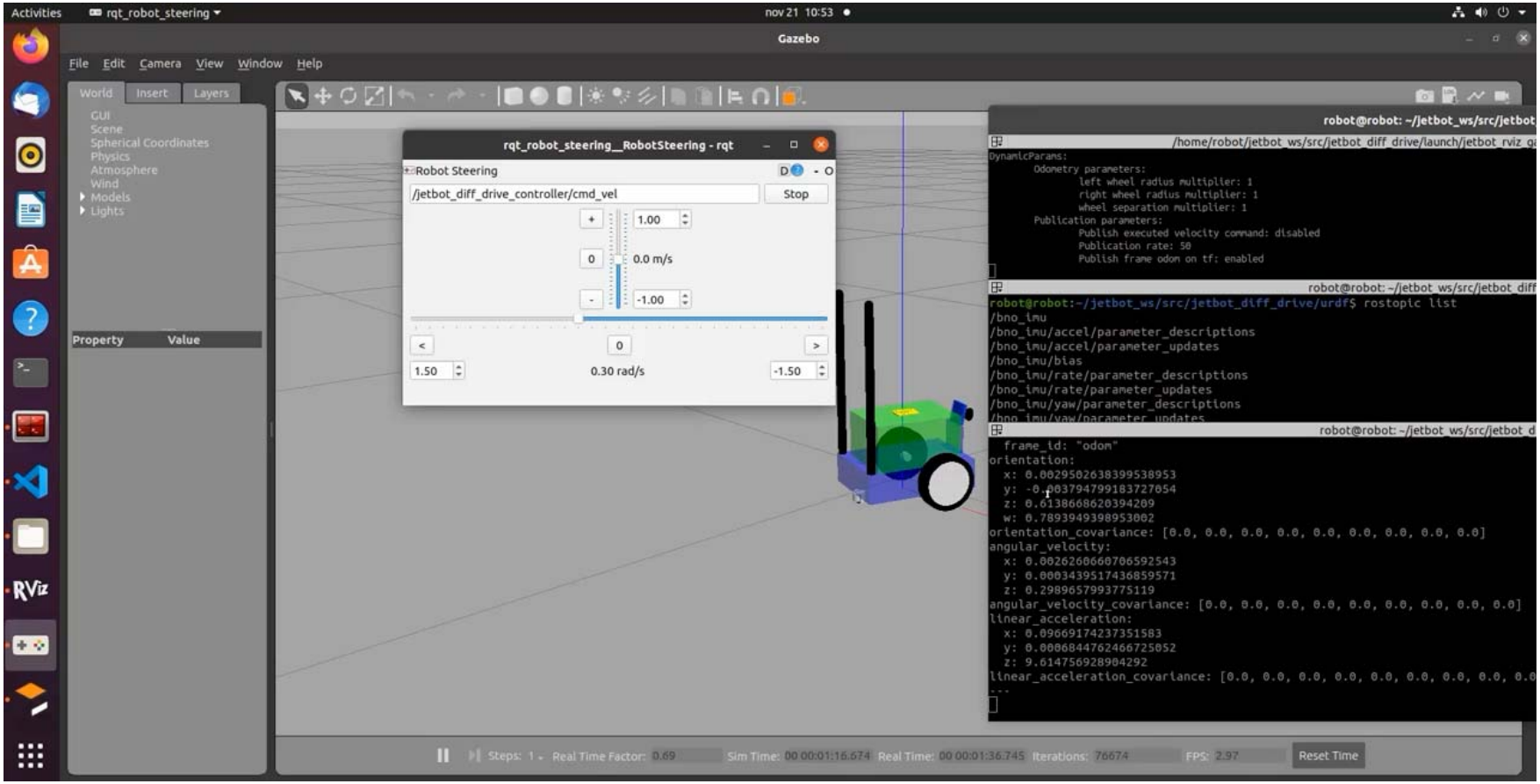
```
#include <ros/ros.h>

#include <sensor_msgs/Image.h>
#include <sensor_msgs/image_encodings.h>

#include <jetson-utils/gstCamera.h>

#include "image_converter.h"


// globals
gstCamera* camera = NULL;

imageConverter* camera_cvt = NULL;
ros::Publisher* camera_pub = NULL;


// aquire and publish camera frame
bool aquireFrame()
{
	float4* imgRGBA = NULL;

	// get the latest frame
	if( !camera->CaptureRGBA((float**)&imgRGBA, 1000) )
	{
		ROS_ERROR("failed to capture camera frame");
		return false;
	}

	// assure correct image size
	if( !camera_cvt->Resize(camera->GetWidth(), camera->GetHeight(), IMAGE_RGBA32F) )
	{
		ROS_ERROR("failed to resize camera image converter");
		return false;
	}

	// populate the message
	sensor_msgs::Image msg;

	if( !camera_cvt->Convert(msg, imageConverter::ROSOutputFormat, imgRGBA) )
	{
		ROS_ERROR("failed to convert camera frame to sensor_msgs::Image");
		return false;
```

```
// node main loop
int main(int argc, char **argv)
{
	ros::init(argc, argv, "jetbot_camera");

	ros::NodeHandle nh;
	ros::NodeHandle private_nh("~");

	/*
	 * retrieve parameters
	 */
	std::string camera_device = "0";	// MIPI CSI camera by default

	private_nh.param<std::string>("device", camera_device, camera_device);

	ROS_INFO("opening camera device %s", camera_device.c_str());


	/*
	 * open camera device
	 */
	camera = gstCamera::Create(camera_device.c_str());

	if( !camera )
	{
		ROS_ERROR("failed to open camera device %s", camera_device.c_str());
		return 0;
	}


	/*
	 * create image converter
	 */
	camera_cvt = new imageConverter();

	if( !camera_cvt )
	{
		ROS_ERROR("failed to create imageConverter");
		return 0;
	}


	/*
	 * advertise publisher topics
	 */
	ros::Publisher camera_publisher = private_nh.advertise<sensor_msgs::Image>("raw", 2);
	camera_pub = &camera_publisher;
```

Conclusion

- Able to develop code that allows jetbot to travel around an area autonomously
- Code uses sensors such as camera and infrared array to distinguish a sleeping person
- Once identified sleeping person is notified and encouraged to wake up

Demonstration


Activities
rqt_robot_steering
nov 21 10:53
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
rqt_robot_steering__RobotSteering - rqt
Robot Steering
/jetbot_diff_drive_controller/cmd_vel
Stop
1.00
0.0 m/s
-1.00
1.50
0.30 rad/s
-1.50
robot@robot: ~/jetbot_ws/src/jetbot
DynamicParams:
Odometry parameters:
left wheel radius multiplier: 1
right wheel radius multiplier: 1
wheel separation multiplier: 1
Publication parameters:
Publish executed velocity command: disabled
Publication rate: 50
Publish frame odom on tf: enabled
robot@robot:~/jetbot_ws/src/jetbot_diff_drive/urdf$ rostopic list
/bno_imu
/bno_imu/accel/parameter_descriptions
/bno_imu/accel/parameter_updates
/bno_imu/bias
/bno_imu/rate/parameter_descriptions
/bno_imu/rate/parameter_updates
/bno_imu/yaw/parameter_descriptions
frame_id: "odom"
orientation:
x: 0.0029502638399538953
y: -0.003794799183727054
z: 0.6138668620394209
w: 0.7893949398953002
orientation_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
angular_velocity:
x: 0.0026260660706592543
y: 0.0003439517436859571
z: 0.2989657993775119
angular_velocity_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
linear_acceleration:
x: 0.09669174237351583
y: 0.0006844762466725052
z: 9.614756928904292
linear_acceleration_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
Steps: 1
Real Time Factor: 0.69
Sim Time: 00 00:01:16.674
Real Time: 00 00:01:36.745
Iterations: 76674
FPS: 2.97
Reset Time

Questions?