UNIVERSITY OF CALIFORNIA, DAVIS
DEPARTMENT OF AEROSPACE AND MECHANICAL ENGINEERING
EME185A A02 W2016

The Winery Floor Cleaning Robot Team

Author:
Erick CAMPOS
Kyle CUBBA
Jerry LI
Dai TRAN

Supervisor:
Jason MOORE
Matthew LEFORT
Farhad GHADAMLI

February 25,2016

Overview

Why a Cleaning Robot for a Winery ?

- Currently, it takes six volumes of water to make one volume of wine, of which is entirely used in cleaning.
- Our sponsor requires an autonomous robot that cleans the crush pad of grape skins, stems, juice, residue, etc. with minimal water use.
- A prototype build by a previous Senior Design Team exists (the WEINbot) though it is not functional at the moment.

Figure: Area of Cleaning

Sponsor Needs

- Minimal water use
- Cleans floor completely
- Autonomous operation
- Avoids static obstacles (walls) and dynamic obstacles (people walking in front)
- Safe to handle
- Easy to clean
- Battery powered and cleans entire area without being required to be charged
- Know when waste container is full
- Know when water container is empty

Figure: Winery Bot

SCOPE OF OUR PROJECT

- Improving upon previous Senior Design Project
- Focus on automation of existing prototype:
  - Mapping
  - Localization
- Meet sponsors needs of:
  - Autonomous operation
  - Safety/Obstacle detection
- Design Specifications Met:
  - Clean 800 sq. feet within 1 hour
- Assumptions about the WEINbot:
  - It solves water usage problem (e.g. uses less water than hosing down grape waste into the drain)
  - The WEINbots brush/conveyor system picks up grape waste effectively
  - Batteries are large enough to clean crush pad on one charge

MAPPING

- What is mapping?
  - In robotics, mapping is the process of generating a floor plan of the accessible area that a robot can move to
- Why is mapping important?
  - Allows a robot to determine location based needs.
  - Obstacle locations
  - What has been cleaned
  - Where it can travel
  - Known location at any given time

How We Plan on Mapping

- A map of the crush pad will be pre-programmed.
  - Wall boundaries
  - Permanent static obstacles
  - Landmarks
  - Parking zones for winery equipment
- Develop an occupancy grid
  - A graph paper like grid space which contains locations of objects in its cells.
  - Locations will be indicated by values of 0 (clear) 1 (obstacle)
  - Each grid will be the size of the robot + turning space

LOCALIZATION

- What is localization?
  - Localization refers to the robots position within its own occupancy grid.
- Why is localization important?
  - Allows robot to obtain necessary information while moving
  - Distance to obstacles
  - Where the robot has traveled

HOW WE PLAN ON LOCALIZING THE ROBOTS POSITION ?

- Dead-reckoning
  - Use the robots speed, orientation, and time traveled to estimate the displacement
  - Speed is determined by input voltage to the motor
  - Orientation is determined using a Digital compass
- Landmarks
  - Pre-programmed locations inside of occupancy grid
  - Landmarks send signal to robot indicating a precise location inside of the occupancy grid.
  - Confirms robot position inside occupancy grid with high accuracy.
- RFID (Radio-frequency identification) tags
  - Markers placed on ground
  - Communicate to receiver that will be mounted to the WEINbot

Safely Avoid Obstacles

- The WEINbot can avoid obstacles by reading input from sensors.
  - LIDAR mounted on front of robot
    - Provides distance data
    - Can be used to stop robot if object is detected within a specified threshold
  - Infrared Sensors
    - Similar to LIDAR
    - Several sensors will point in all directions around the robot
  - Bumper (pressure sensitive switch)
    - Mounted to front of robot
    - Will shut down robot completely i triggered

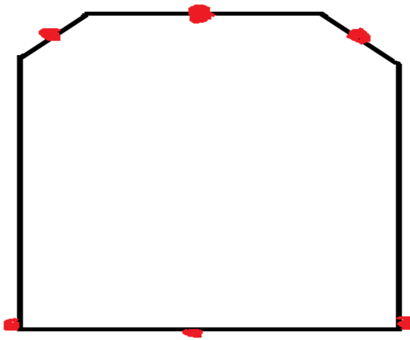

Figure: Sensors Positioning

ROBOT OPERATION

- Occupancy grid size
  - x50 3.2x5sqft. grids
  - Robot will move randomly until hit an obstacle where the turning direction will depend on:
    - Available directions to move
    - Density of dirty floor grids
  - RFID density depends on dead reckoning error:
    - Sample average straight line error for 1 minute n=30+
    - Find 95% confidence error/meter traveled
    - Set maximum allowable error before location reset (landmark)

Schedule of Primary Tasks

By end of Winter Quarter	March	April	May
• Continue Beaglebone reading • Research on RFID tags and Dead Reckoning methods • Finish wiring WEINbot • Find RFID density • Learn about programming sensor and begin coding	• Begin programming motors (for wheels, conveyor, etc.) • Calibrate wheels • Determine robot kinematics • Finalize area that robot will be bounded to	• Add additional sensors for object and landmark detection • Place landmarks at winery • Program sensors • Finish programming motors, servo	• Program map into robot • Test run • Improve as necessary

Table: Schedule of Primary Tasks

Estimated Cost of Project Materials

Item	Cost	Quantity	Subtotal
Infrared Sensors	15	7	105
RFID Tag	2	20	40
RFID Reader	35	1	35
Cable	10	1	10
Total			**190**

Table: Estimated Cost of Project Materials

Q&A

```python
#This is the demonstration of using analog sensors with PWM actuator
import Adafruit_BBIO.GPIO as GPIO
import time
import Adafruit_BBIO.PWM as PWM
import Adafruit_BBIO.ADC as ADC
#PWM.start(channel, duty, freq=2000, polarity=0) #syntax
#Make variable for easier assigning pin number
blueRGB    = "P8_15"
greenRGB   = "P8_16"
redRGB     = "P8_17"
irTransmitter = "P9_14"
irReceiver = "P9_37"
lightResistor = "P9_40"

GPIO.setup(blueRGB, GPIO.OUT) #RGB LED
GPIO.setup(greenRGB, GPIO.OUT) #RGB LED
GPIO.setup(redRGB, GPIO.OUT) #RGB LED
GPIO.setup(irTransmitter, GPIO.OUT) #Infrared transmitter Pin
GPIO.setup(irReceiver, GPIO.IN) #Infrared receiver Pin
GPIO.setup(lightResistor, GPIO.IN)
ADC.setup()

def blinkLEDs(): #this fuction will blink RGB LEDs
    GPIO.output(blueRGB, GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(greenRGB, GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(redRGB, GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(blueRGB, GPIO.LOW)
    time.sleep(0.1)
    GPIO.output(greenRGB, GPIO.LOW)
    time.sleep(0.1)
    GPIO.output(redRGB, GPIO.LOW)
```