Emergency Stop Feature

DCRC Project

Safety Course Level 2

Background:

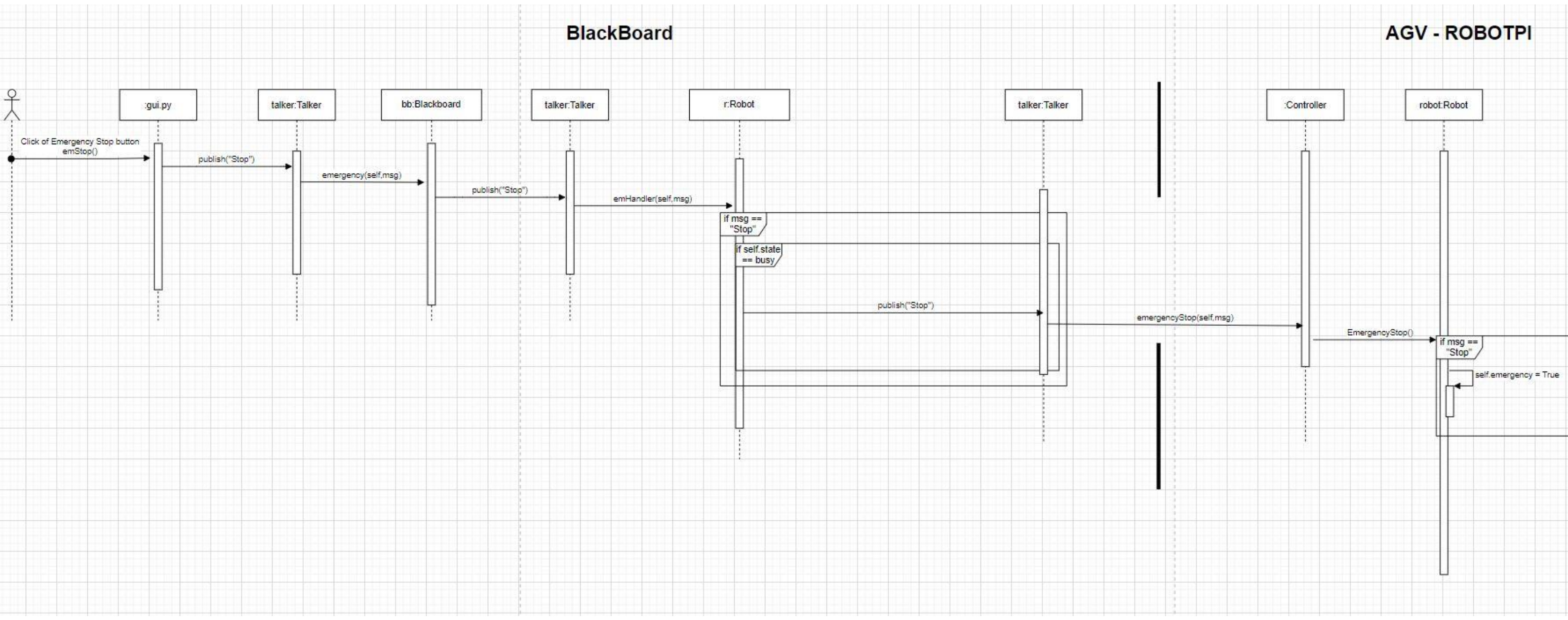
The DCRC Project consists of a fleet manager and colab software for locating AGVs. How it works is that there are 2 raspberry pi's on each robot, one for the colab localization software and one pi for the robot itself. The fleet manager is a program called the blackboard. The purpose of the blackboard is to assign tasks to the robot. It has a graphical user interface so that a person can assign a task and the program will assign that task to a robot from the fleet that is not busy at the moment. The blackboard is run on a laptop not on a pi and is connected to the colab pi with an ssh connection, which means they can communicate with topics because of the ssh. The blackboard works with ros and the robot pi software is in ros2 so besides the ssh connection we need a bridge between ros and ros2 to make them work with topics. The colab software on the colab pi is connected to the robot with TCP server so that the robot can receive its current location. The robot code on the robot pi is solely dedicated to receiving messages over topics and tcp and making the robot move to a goal and perform the robot movement functionalities.

Current situation:

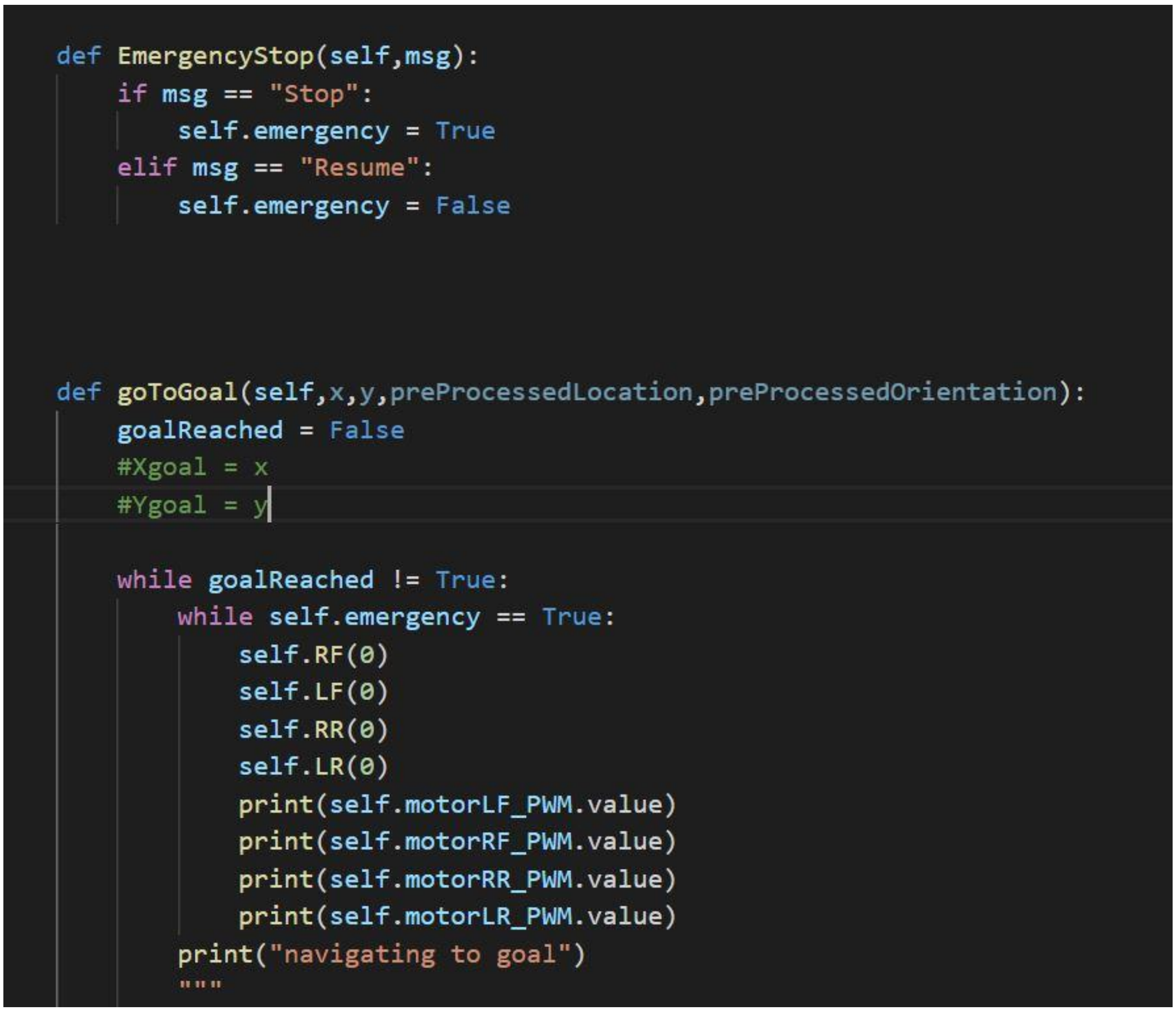
In the previous section I explained the best case scenario where everything works. The current situation is that the ros bridge, which was assigned to another student is not working meaning that there is no way to give the robots a task to execute. The SSH connection also doesn't work as it is supposed to as it disconnects randomly. The robot code on the robot pi is also not working properly as the GPIO library that is used to tell the motors to drive doesn't work as it gives an error that it is not being run on a raspberry pi, which means that we can't have ros if we want to have the GPIO work. It is a choice between the GPIO or ROS in a situation in which we need both. These two paragraphs explaining the project and current status of it will be omitted in the next assignments I submit.

Execution:

The goal of the emergency stop feature as planned and implemented is as follows:



The user clicks the button on the graphical user interface and that button publishes a message "Stop" on a topic to the blackboard script. The blackboard is subscribed to that topic, receives the message and sends a new message on another topic to the robot script of the blackboard, which checks the message if it is "Stop" or "Resume" and if it is "Stop" it passes the message through another topic to the Controller node of the robot pi code. The Controller receives the message and calls a method on the Robot class to change the emergency variable to TRUE. What happens is that inside the method, which is used for driving the robot to the goal, there is a while cycle that states that until the emergency variable is false we make the motors stay idle. The idea is that if the variable is FALSE then we don't even go into that loop and if it is TRUE then we stay inside it, which is exactly what we want. A screenshot of the last part of the code that is in the Robot class on the robot pi can be seen below. There are no screenshots on the other parts of the feature as it is only topics and passing the message, which is too basic.



```
def EmergencyStop(self,msg):
    if msg == "Stop":
        self.emergency = True
    elif msg == "Resume":
        self.emergency = False



def goToGoal(self,x,y,preProcessedLocation,preProcessedOrientation):
    goalReached = False
    #Xgoal = x
    #Ygoal = y

    while goalReached != True:
        while self.emergency == True:
            self.RF(0)
            self.LF(0)
            self.RR(0)
            self.LR(0)
            print(self.motorLF_PWM.value)
            print(self.motorRF_PWM.value)
            print(self.motorRR_PWM.value)
            print(self.motorLR_PWM.value)
        print("navigating to goal")
        """
```

Result:

Because of all the problems with the communication and driving the robot, the full functionality can’t be tested. What was done, was actually only a small test to prove the concept. The test consists of only the robot pi code run on a raspbian because as I said earlier, the library for the motors don’t work if there is ros on the pi. The test case is that we give an X and Y goal to the robot, it starts navigating to it and we trigger the change of the variable called “emergency” to become TRUE after a few seconds. The code gets into the while cycle and the robot stops. Unfortunately, the full functionality can’t be tested and that doesn’t depend on me as the ros bridge and robot navigation weren’t my assignments. I tried my best to at least prove that the key part of the code works. The proof of the result can be seen in the videos attached to the email. The videos show how I establish an ssh connection to the raspberry and I run the test script which makes the robot goal to a predefined goal and after a few seconds the variable is set to True and it stops. As a conclusion I can say that once the bridge and GPIO problems are solved the feature can be implemented and put into production instantly.

All the code for the feature can be found on: https://github.com/fontysrobotics/DCRC