**Technical Report – User Interface for WiFi Communication Between Laptop and Raspberry Pi**

**Requirements:**

Write a program so the user can enter a command and value (e.g. “Go forward 10cm” or “Go left 30 degrees”), then press a button which sends that information via WiFi to the Raspberry Pi attached to the robot, which then executes the command. A desirable feature would also allow for multiple commands to be sent at once to be executed consecutively. In addition, the robot should be able to send values (e.g. the value from the ultrasonic sensor ) back to the program upon request of the user or at regular intervals.

**Class Analysis:**

There are five classes, ClientServer, Client, CommandPanel, ChoosePanelNumber and RobotUser:

The ClientServer contains methods for reading from and writing to the input and output streams.

The Client class extends ClientServer and contains a Client constructor which creates the socket to communicate with the Raspberry Pi. It also contains methods for handling events from the GUI.

The CommandPanel extends JPanel and contains a constructor which creates the drop down menu and textfield so the user can input commands. It also event handling methods which check inputs and if valid passes to methods in the Client class to communicate with the Raspberry Pi.

The ConditionPanel class extends JPanel and contains a constructor which contains a checkbox, a drop down menu containing commands and two text fields for inputs so the user can choose what the robot should do if there is an object within a certain distance. It contains methods to get the selected item in the drop down menu or the input in the textfields so that they can be called in the handleConditionEvent() method.

The ChoosePanelNumber class extends JPanel and contains a constructor which creates a drop down menu and button to choose how many instances of CommandPanel to create and add to the GUI to allow for multiple commands to be sent at once. It also contains event handling methods which creats and adds instances of CommandPanel to ChoosePanelNumber when the ‘Ok’ button is pressed and calls methods in CommandPanel and ConditionPanel if the ‘Go!’ button is pressed.

The RobotUser class extends JFrame and contains a constructor which adds an instance of ChoosePanelNumber and ConditionPanel in addition to an ultrasonic sensor JButton and its Action Listener. It contains the actionPerformed() method which calls a method from the Client class to read the ultrasonic sensor value from the input stream. The main method in this class creates a new instance of Client and RobotUser, sets the size of the window and exits the program if the window is closed.

**Class Diagram:**

ChoosePanelNumber

* numbers[]: String
* numberOfPanels: JComboBox
* prompt: JLabel
* ok: JButton
* go: JButton
* mainPanel: JPanel
* commandPanels: JPanel
* panelArray: List<CommandPanel>

+ actionPerformed(ActionEvent)

+handleEvent()

RobotUser

* panel: ChoosePanelNumber
* ultra: JButton
* send: JButton
* display: JTextArea

+ actionPerformed(ActionEvent)

+addBorder(JPanel, String)

+main(String[])

ConditionPanel

* commands[]: String
* choices: JComboBox
* checkbox: JCheckBox
* distanceInput: JTextField
* valueInput: JTextField
* ifLabel: JLabel
* unitLabel: JLabel
* labelUse: JLabel
* labelTurn: JLabel
* labelUse: JLabel

+ getTriggerDistance(ConditionPanel): String

+getCommand(): String

+getValue(): String

+handleConditionEvent(ConditionPanel)

CommandPanel

* instructionNo: int
* commands[]: String
* choices: JComboBox
* input: JTextField
* labelTurn: JLabel
* labelMove: JLabel
* labelUse: JLabel
* commandNo: JLabel
* maxMove: int
* maxAngle: int

+ itemStateChanged(ItemEvent)

+handleUserEvent()

+checkMove(int): boolean

+checkAngle(int): boolean

JFrame

Client

- socket: Socket

- IP: String

- port: int

+ handleNetworkEvent(String)

+getValue(): String

ClientServer

+ readFromSocket(Socket)

+writeToSocket(Socket)

**Implementation and testing:**

The code has been written to implement the above classes and methods and can be found on GitHub at legorovers/PiRover. The testing will test that the program reacts as expected to user input and also gives expected outputs. It will also test that the user interface stays responsive throughout.