piRover Builds with K2

Project 02 – User Move Rev 1.0

Goal:

Your task is to create the User Move solution. The solution will allow you to "drive" the piRover by entering move commands from the keyboard. This project requires you to use code and coding patterns completed during the User Blink and Motor Control Intro activities. You will use code similar to User Blink to input and process the user's drive commands. You will use the move functions created in Motor Control Intro to implement the drive actions specified by the user.

Additional credit will be awarded for implementing new drive behaviors and loop control. Focus on basic functionality first. Most of the assessment points for the solution development are assigned here. See the grading rubric at the end of this document for detail. Extend the solution to include new drive and loop control as time permits.

This is the final assessment for Sprint 2. Do your best and do you own work. Do not assist others. Use comments to identify issues with your code.

Prerequisites:

This assessment requires content and code solutions from the following:

- User Blink
- Motor Control Introduction

Performance Outcomes:

1. Input drive commands from a user's keyboard.
2. Process drive commands using selection structures.
3. Call move functions based on user input.
4. Extend move functions using existing move patterns.
5. Control program flow

Resources:

1. See prerequisite lessons

Materials:

1. piRover

piRover Builds with K2

Part 1 – Set Up

1. Prepare your workspace for this project.
    a. Connect to your piRover using VNC. Access your piRover folder and launch VS Code.
    b. Create a **08.UserMove** directory.
    c. Create a new **user_move.py** file in the 08.UserMove directory. This is the solution file for this project.
    d. Locate the **user_blink.py** code in the 02.UserBlink directory. Copy this file the new 08.UserMove directory. You will reference this file when creating input and selection code.
    e. Locate the **motor_control_intro.py** code in the 07.MotorControlIntro directory. Copy this file the new 08.UserMove directory. You will implement drive operations for this project by copying drive functions defined in this file to your new solution file.
    f. Both user_blink.py and motor_control_intro.py files will be included in your final project submission.

Part 2 – Basic Drive Control

1. Copy the drive code from the motor_contro_intro.py file to your new user_move.py file. Modify the docstring to include the new solution title. Be sure to include your name and the date.
2. Run the code to verify that you have basic drive operations implemented including forward, backward, left turn, and right turn. This basic drive code was reviewed in class prior to this project. A turn is defined as having one track stop while having the opposite track move forward.
3. Review the input and selection pattern used in user_blink.py. Replace the list of drive commands called at the bottom of your initial user_move code from step 1 with user input and selection code that implements the following requirements. This represents the "main" code of the solution.
    a. Configure the GPIO.
    b. Welcome the user to the "User Move" solution.
    c. List the valid move commands for the user. Initially these are single key inputs listed in table 1 on the following page. **Recall that the user will need to press the Enter key after the command key due to the limitations of using Python's input() function.**

Reacting to key presses (no Enter key required) is beyond the scope of the class at this time.

Table 1-User Commands

User Command	Move	Comment
W	Forward	piRover move forward for 2 seconds
A	Left	piRover turns left for 2 seconds
D	Right	piRover turns right for 2 seconds
X	Backward	piRover move backward for 2 seconds

d. Create an infinite loop that captures the user's input and calls the appropriate move function. Use a 2 second delay for all moves.

e. Be sure to allow the user to enter commands in either upper or lower case.

f. Inform the user if they have entered an invalid move command.

4. Test your code with all possible user inputs. Be sure you have your piRover propped up in preparation for track motion.

5. Note that most of the points in this assessment are assigned to the steps above. Be sure this basic code is functional prior to moving on to the next extension sections.

Part 3 – Extending Drive Control

1. Use the GPIO output patterns used in existing move functions to create two new move functions – left_rotate() and right_rotate(). A rotate turn is defined as having one track move backwards while the other track moves forward. Use the sec parameter in these new functions just as with prior move functions.

2. Assign the user input "AA" to rotate left and "DD" to rotate right. Modify input and selection code as required to enable these new commands and drive functionality.

3. Modify the user message in the main code to include "AA" and "DD" in the list of valid commands.

4. Test your code.

piRover Builds with K2

Part 4 – Controlling Move Duration

1. Research the requirements for getting an integer value from the keyboard using this link.

   https://www.askpython.com/python/examples/python-user-input

2. Modify your code to prompt first for a move and then for a move duration. For example, your code would also display a message like "How long would you like to move?" or "Please enter move duration".

3. Capture this delay value from the user and assign to the move function being called. Be sure to reference the information from the link provided in step 1.

4. Test your code.

Part 5 – Controlling Exit

1. Add "Q" as a valid user input representing Quit. Add a message telling the user to enter "Q" to quit the application.

2. Modify the main control loop so that if the user's command is Q, the loop terminates, and the program ends. Add GPIO.cleanup() as the last line of code. (Note: if you implemented part 4, then your program may ask the user how long they would like to Quit. While this makes no sense, it is okay for this solution. Other options will be presented in the class discussion.)

User Move – Final

1. Table 2 indicates the functionality of the final solution where **s** is the move delay entered by the user.

Table 2 - User Commands - Final

User Command	Move	Comment
W	Forward	piRover moves forward for **s** seconds
A	Left	piRover turns left for **s** seconds
AA	Rotate Left	piRover rotates left for **s** seconds
D	Right	piRover turns right for **s** seconds
DD	Rotate Right	piRover rotates right for **s** seconds
X	Backward	piRover moves backward for **s** seconds
Q	Quit	Program exits. See note on move delay.

piRover Builds with K2

Submission:

1. Review the grading rubric below. Partial credit is awarded. Use comment lines to identify issues.
2. Zip your solution file with the other related files as UserMove.zip. Your zip file must contain.
    a. **user_move.py**
    b. user_blink.py
    c. motor_control_intro.py
3. Submit your zip file to the Moodle link by the deadline specified.

P02 - User Move - Project Evaluation			
Part	**Description**	**Points**	**Score**
1	user_move.py file created and submitted as .zip	2	2
	motor_control_intro file included as reference	2	2
	user_blink file included as reference	2	2
	Solution contains initialization and move functions copied from motor_control_intro	3	3
2	GPIO is initialized	4	4
	Message output to terminal listing valid move commands	4	4
	Move command input obtained - upper and lower case	4	4
	User move is matched to possible moves	4	4
	Message to user if invalid move is entered	2	2
	Valid move calls appropriate move function	4	4
	Solution contains valid Python code with correct syntax	4	4
	SUBTOTAL	35	35
3	New rotate functions created	3	3
	New rotate functions implemented	3	3
4	User prompted for move delay	3	3
	User's move delay assigned to function parameter	3	3
5	Quit command ends program	3	3
	SUBTOTAL	15	15
	TOTAL	50	50