ECEN 4213
Embedded Computer System Design

Lab 3: Robot Motion Control

Instructor: Dr. Weihua Sheng, weihua.sheng@okstate.edu

TA: Zhanjie Chen, zhanjie.chen@okstate.edu

Claudia Pauyac, cpauyac@okstate.edu

Fall 2025

I. Lab 2 Due Date and Submission

II. Lab 3 Introduction

III. Lab 3 Due Date and Submission

Due date

- Lab demonstration:
  - ✓ no later than 7: 20 pm, September 23, 2025 (Tuesday Session)
  - ✓ no later than 7: 20 pm, September 24, 2025 (Wednesday Session)
  - ✓ no later than 5: 20 pm, September 26, 2025 (Friday Session)

- Lab report:
  - ✓ no later than 11: 59 pm, September 23, 2025 (Tuesday Session)
  - ✓ no later than 11: 59 pm, September 24, 2025 (Wednesday Session)
  - ✓ no later than 11: 59 pm, September 26, 2025 (Friday Session)

Submission (in a ZIP file)

- Lab report (Word or PDF file)
- Your code: *Lab2EX1.cpp*, *Lab2EX2.cpp*

I. Lab 2 Due Date and Submission

II. Lab 3 Introduction

III. Lab 3 Due Date and Submission

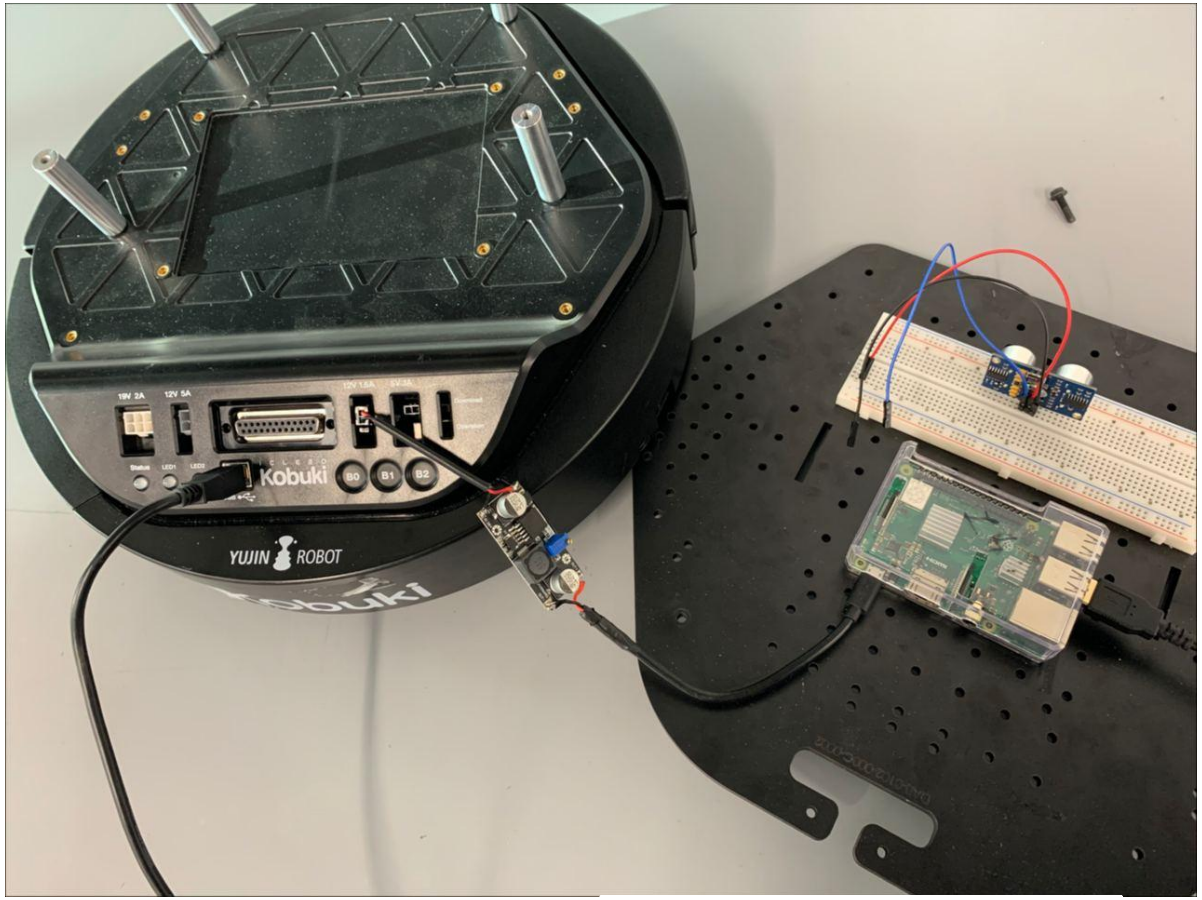
II. Lab 3 Introduction

Lab 3 Objectives

- Be familiar with Kobuki robot
- Be able to control Kobuki with joystick
- Be able to control Kobuki with joystick via WiFi router

Kobuki

- PC connection: USB interface for communication
- Charging & Power Supply: Kobuki can be charged via a DC input, powering both itself and optionally a microcontroller via a step-down converter
- Bumper sensors: Located at the front in three regions – left, center, right – for collision detection
- Cliff sensors: Detect sudden drops (like stairs), also divided into left, center, and right
- Wheel Drop sensors: Detect if either left or right wheel is lifted off the ground



Supplemental document: ***Kobuki_User_Guide.pdf***

Powering the Raspberry Pi via Power Bank

- In this setup, the Raspberry Pi is mounted on top of the Kobuki robot and is powered using a portable power bank.
- The power bank provides a stable 5V output through USB, which is exactly what the Raspberry Pi needs.

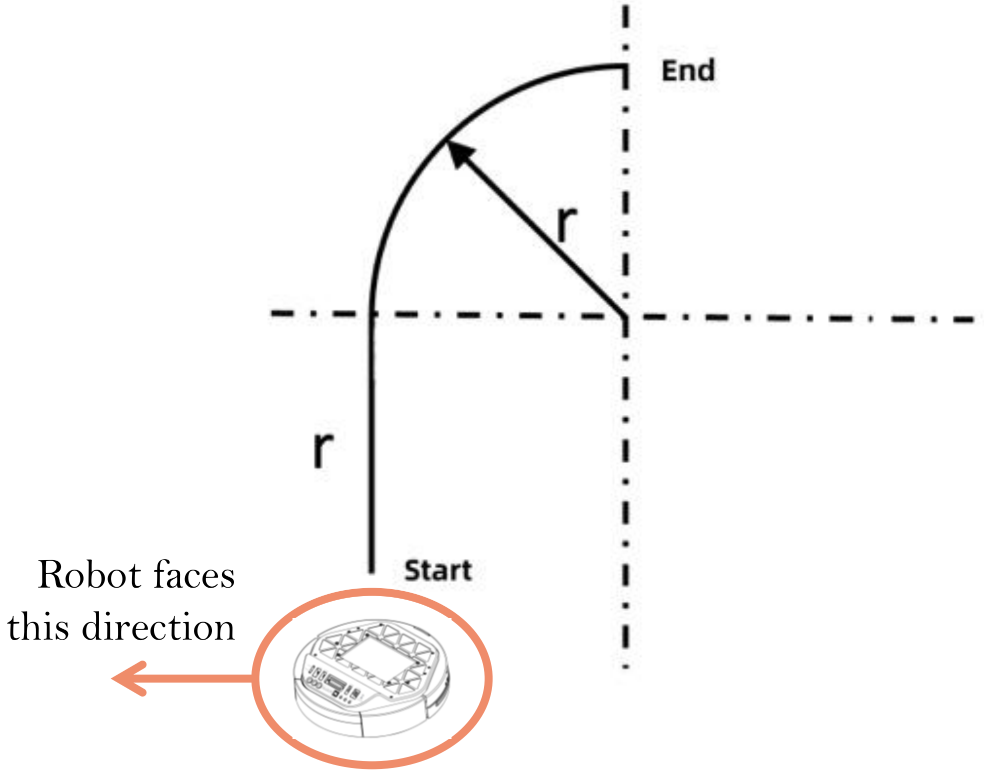
Why use a power bank here?

- It makes the Raspberry Pi independent from the Kobuki's internal power supply.
- It provides safety: if something goes wrong with wiring, the Pi is protected because the power bank has built-in safety circuits.
- It adds mobility: you can test your Pi and robot anywhere without being tied to an outlet.

Exercise 1 – Serial communication and Motion Control

Explanation: This exercise teaches how to program the Kobuki using serial communication.

- Objective: Write a program to control Kobuki's motion along a curved path with a 50cm radius ($r = 50cm$)
- Connection: Connect one end of the USB cable to Kobuki's serial port, the other to Raspberry Pi
- Path execution: Move the robot in an arc using velocity commands. Add a stop (*movement(0,0)*) between commands to ensure smooth transitions
- Demo: Show the robot executing the programmed motion to the lab instructor



Supplemental document: ***Kobuki Protocol Specification.pdf***, ***Serial Library_WiringPi.pdf***

Byte Stream Format and Motion Command Packet Structure

Explanation: How to send motion commands using a custom byte stream

Packet format:

- **Headers (*b_0*, *b_1*):** Start of frame indicators
- **Payload length (*b_2*) & Sub-payload header and length (*b_3*, *b_4*):** Define content length and command type
- **Speed (*b_5*, *b_6*) and radius (*b_7*, *b_8*):** Sent as 2-byte values each, in mm/s and mm, respectively
- **Checksum:** XOR of all bytes (excluding headers *b_0*, *b_1*) for data integrity

Note: The code example shows how to build this byte array and transmit it using *serialPutchar()*

```c
void movement(int sp, int r){
    //Create the byte stream packet with the following format:
    unsigned char b_0 = ; /*Byte 0: Kobuki Header 0*/
    unsigned char b_1 = ; /*Byte 1: Kobuki Header 1*/
    unsigned char b_2 = ; /*Byte 2: Length of Payload*/
    unsigned char b_3 = ; /*Byte 3: Sub-Payload Header*/
    unsigned char b_4 = ; /*Byte 4: Length of Sub-Payload*/

    unsigned char b_5 = sp & 0xff;  //Byte 5: Payload Data: Speed(mm/s)
    unsigned char b_6 = (sp >> 8) & 0xff; //Byte 6: Payload Data: Speed(mm/s)
    unsigned char b_7 = r & 0xff;   //Byte 7: Payload Data: Radius(mm)
    unsigned char b_8 = (r >> 8) & 0xff;    //Byte 8: Payload Data: Radius(mm)
    unsigned char checksum = 0;     //Byte 9: Checksum

    //Checksum all of the data
    char packet[] = {b_0,b_1,b_2,b_3,b_4,b_5,b_6,b_7,b_8};
    for (unsigned int i = 2; i < 9; i++)
        checksum ^= packet[i];

    /*Send the data (Byte 1 - Byte 9) to Kobuki using serialPutchar (kobuki, );*/

    /*Pause the script so the data send rate is the
    same as the Kobuki data receive rate*/

}
```

Supplemental document: ***Kobuki Protocol Specification.pdf – Page 1***

Structure of ByteStream (4 fields)

Name	Header 0	Header 1	Length	Payload	Checksum
Size	1 Byte	1 Byte	1 Byte	N Bytes	1 Byte
Description	0xAA (Fixed)	0x55 (Fixed)	Size of payload in bytes	Described below	XOR'ed value of every bytes of bytesream except headers

1 Byte → 8 bits → **0 0 0 0** **0 0 0 0** → 0x**00** (hex format)

Base Control application

	Name	Size	Value	Value in Hex	Description
Header	Identifier	1	1	0x01	Fixed
Length	Size of data field	1	4	0x04	Fixed
Data	Speed	2			in mm/s
	Radius	2			in mm

Length field: 1 byte

- This field counts the number of bytes in the payload only (not header, not checksum).
- In the example frame from your skeleton code:

  b0 b1 b2 b3 b4 b5 b6 b7 b8 b9

  The payload is everything between **b2** and **b9** (Checksum byte)

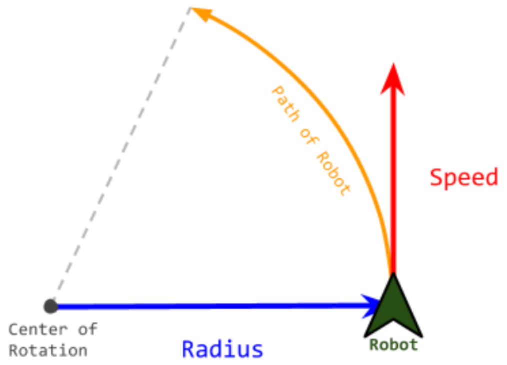
Payload field: N bytes

- Since the purpose of the serial communication is to control the wheel motors (Base Control command), the payload size is **N = 6 bytes** (according to the documentation).
- The bytes to be configured are related to the *header* (or identifier), *length* (or size of *data*) and *data* (2 bytes for speed and 2 bytes for radius)

Understanding Movement Commands and Velocity Representation

Explanation: Mathematics behind robot motion

- Translation + Rotation:
  - Motion is defined via Speed (mm/s) and Radius (mm)
  - The radius determines the curvature of the path. For straight movement: $radius = 0$



Velocity Representation

But actual value of Speed field is little bit different. Here is conversion table.

Motion	Speed(mm/s)	Radius(mm)
Pure Translation	Speed	0
Pure Rotation	w[i] * b[ii] / 2	1

Translation + Rotation	Speed * (Radius + b[ii] / 2) / Radius, if Radius > 1 Speed * (Radius - b[ii] / 2) / Radius, if Radius < -1	Radius

i) w is rotation speed of the robot, in [rad/s].
ii) b is *bias* or *wheelbase*, that indicates the length between the center of the wheels. Fixed at 230 mm.

	Name	Size	Value	Value in Hex	Description
Header	Identifier	1	1	0x01	Fixed
Length	Size of data field	1	4	0x04	Fixed
Data	Speed	2			in mm/s
	Radius	2			in mm

<u>Supplemental document</u>: ***Kobuki Protocol Specification.pdf* – Pages 4 & 5**

Exercise 2 – Kobuki Control Using a Gamepad Joystick

Explanation: This exercise teaches how to control the Kobuki robot using a Logitech gamepad.

- The joystick will interface with your code (in *Lab3EX2.cpp*) to issue movement commands via specific button presses.
- Control mapping:
    - D-Pad Up → Move Kobuki forward
    - D-Pad Down → Move Kobuki backward
    - D-Pad Right → Rotate Kobuki 90° clockwise
    - D-Pad Left → Rotate Kobuki 90° counterclockwise
    - Start Button → Stop the Kobuki
    - Logitech (Select) Button → Cleanly close all communication connections

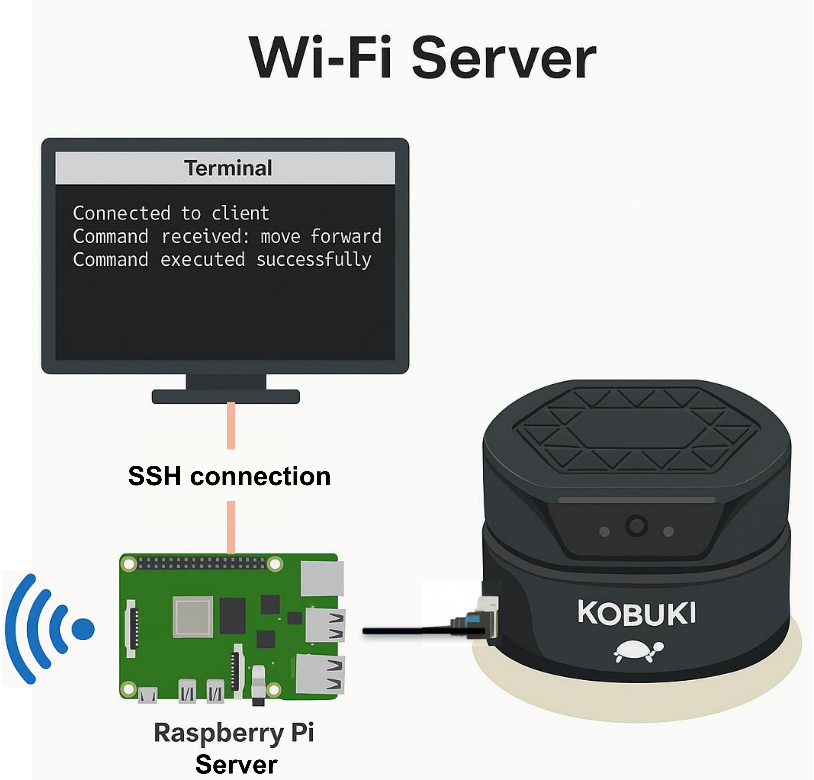
<u>Note</u>: Make sure the files *joystick.cc* and *joystick.h* are in the same directory as *Lab3EX2.cpp*.

🎯 Demonstrate the program to your lab instructor once completed.

Exercise 3 (Part A) – Creating a Wi-Fi Server to Control Kobuki

Explanation: This exercise teaches how to implement a Wi-Fi server program on the RPi.

- The server will receive driving commands over the network from the client and send them to the Kobuki via serial.
- Tasks:
  - Set up a server that accepts socket connections
  - Collaborate with another group acting as the client to test communication
  - Log all incoming data to the terminal for verification (print the output to both terminal screens to verify the connection was created)
  - Ensure all sockets and resources are properly closed when the script ends



🎯 Demonstrate the program to your lab instructor once completed.

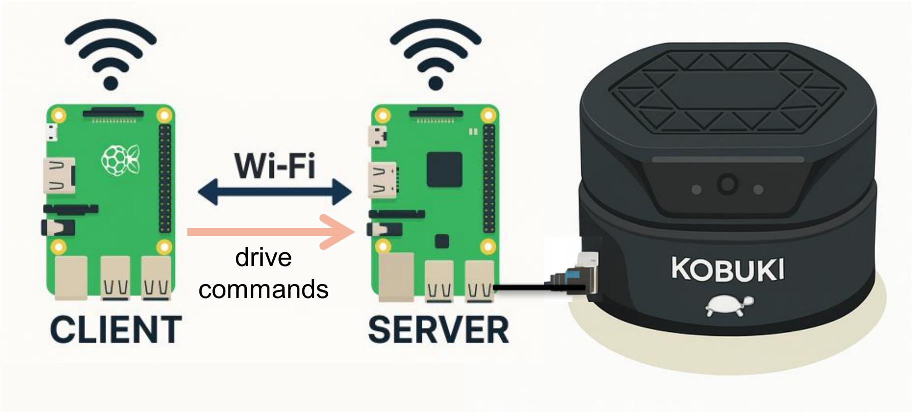
Supplemental document (Socket programming): ***https://beej.us/guide/bgnet/html/split/***

Exercise 3 (Part B) – Creating a Wi-Fi Client to Send Commands

Explanation: This exercise teaches how to build a Wi-Fi client application to send commands to a remote Kobuki server.

- Connect to the server over the same network
- Send drive commands (e.g., "forward", "stop") to the server in the correct format
- Print communication logs in both the client and server terminals to confirm correct transmission

🎯 Demonstrate the program to your lab instructor once completed.



Note: Choose to implement either Part A (server) or Part B (client) – but bonus points are available for doing both, as shown in **Bonus 1**.

Bonus 1: Full Stack Implementation

If your group implements both the client and server sides of Exercise 3, you will receive 3 bonus points.

Bonus 2: Variable Speed & Turning Using the Joystick

- Use the analog joysticks on the Logitech controller:
  - The left joystick controls speed
  - The right joystick controls turning radius
- The joystick raw values range from -32767 to 32767.
- You must scale these to values the Kobuki accepts (e.g., valid speed in mm/s and turning radius in mm).
- If done correctly:
  - Implementing either the client or server with this joystick control earns 4 bonus points.
  - Implementing both sides with variable control earns 6 total bonus points

I. Lab 2 Due Date and Submission

II. Lab 3 Introduction

III. Lab 3 Due Date and Submission

Due date (Three weeks)

- Lab demonstration:
  - ✓ no later than 7: 20 pm, October 14, 2025 (Tuesday Session)
  - ✓ no later than 7: 20 pm, October 15, 2025 (Wednesday Session)
  - ✓ no later than 5: 20 pm, October 17, 2025 (Friday Session)

- Lab report:
  - ✓ no later than 11: 59 pm, October 14, 2025 (Tuesday Session)
  - ✓ no later than 11: 59 pm, October 15, 2025 (Wednesday Session)
  - ✓ no later than 11: 59 pm, October 17, 2025 (Friday Session)

What to submit?

A ZIP file that includes:

- Lab report (Word or PDF file)
  - Supplemental questions
  - Screenshots of your results
  - Pictures of the circuits

- Your code
  - Lab3EX1.cpp, Lab3EX2.cpp, Lab3EX3A.cpp, Lab3EX3B.cpp
  - Lab3BonusA.cpp, Lab3BonusB.cpp

Note: One group, one lab report.

Grading Criteria

The grading criteria is same as listed on the handout; however, if you don't demonstrate your code to TA, then 50% of maximum points are reduced directly.

Office hours

- Tuesday : 4:30 pm – 5:30 pm, Endeavor 350
- Wednesday: 4:30 pm – 5:30 pm, Endeavor 350
- Friday: 3:30 pm – 4:30 pm, Endeavor 350