OSU Mechanical Engineering

Smart Products Laboratory

Robot | Six

ME Course Number
Spring 2019

Table of Contents

List of Figures

1 Overview

This lab we will be programming the different routine to control the Dobot robot.

2 Background

The information below should provide a basic introduction to the concepts.

2.1 Background

The key concepts for this lab are defined and demonstrated below

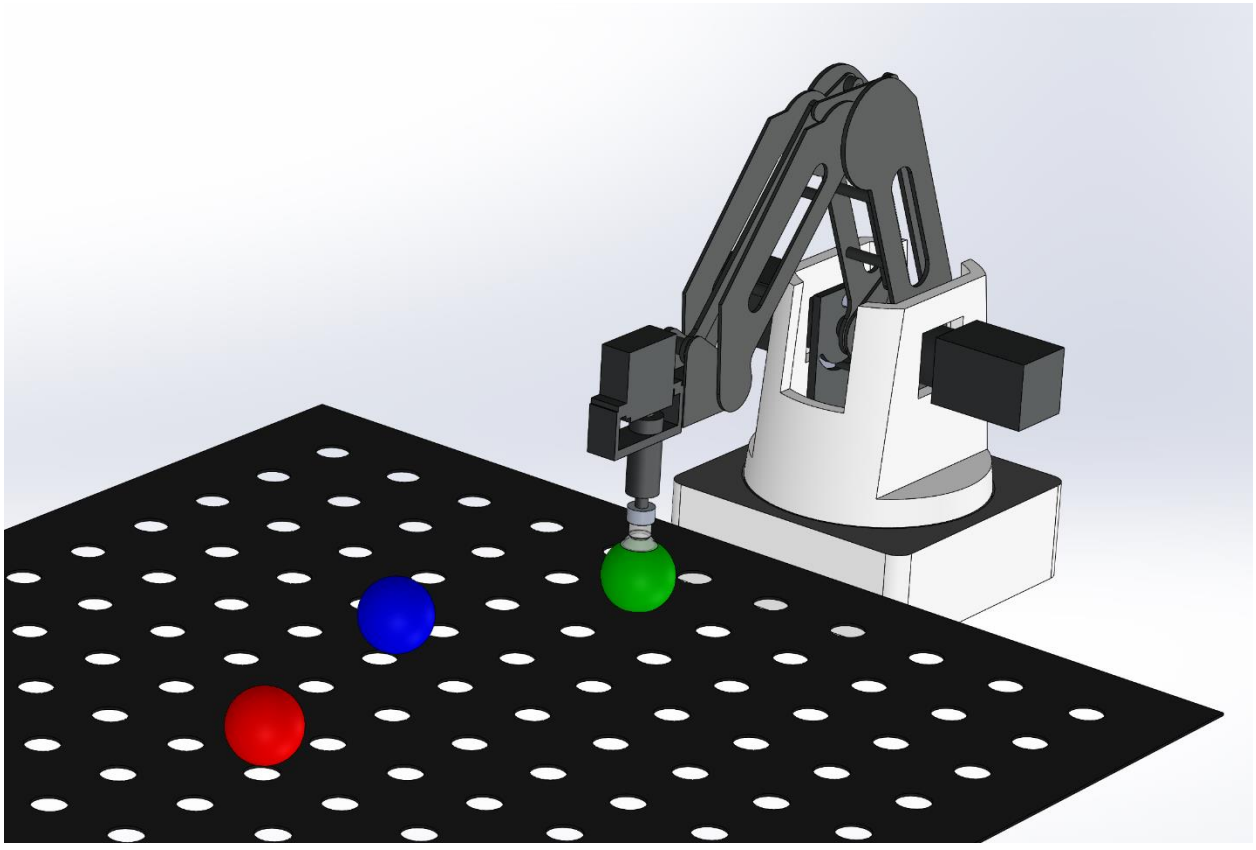
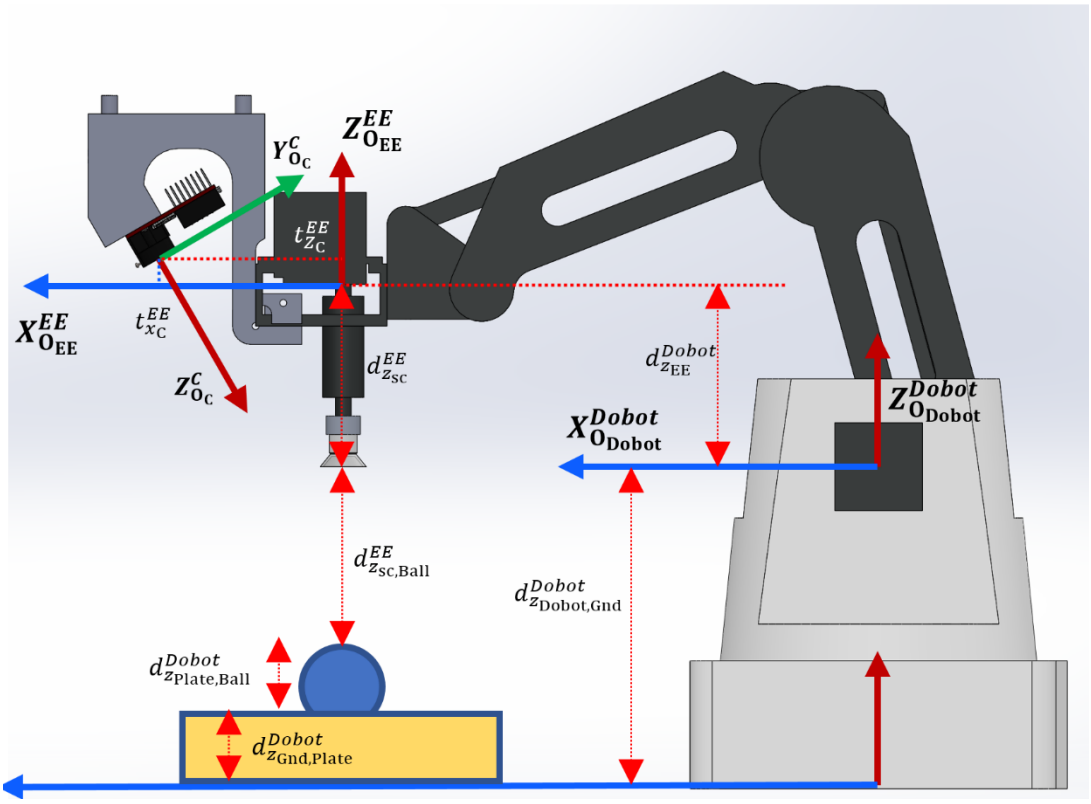


Figure 1 – Robot model



$$d^{EE}_{Z_{SC}} = 76.96\ mm$$

$$d^{Dobot}_{Z_{Dobot,Gnd}} = 139.7\ mm$$

$$d^{Dobot}_{Z_{Plate,Ball}} = 37.59\ mm$$

$$d^{Dobot}_{Z_{Gnd,Plate}} = \textit{will be given, assume 0 for the lab}$$

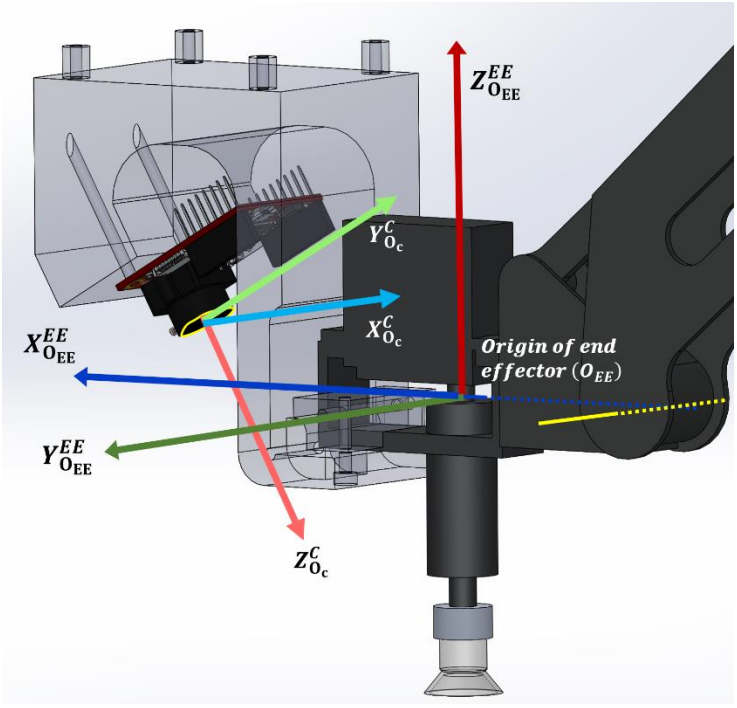


Figure 2 – Robot coordinate frames

3 References & Resources

3.1 References

The following references may be helpful to complete the lab.

1. https://en.cppreference.com/w/cpp/utility/functional/function
2. https://www.dobot.cc/dobot-magician/specification.html
3. https://www.dobot.cc/download/dobot-communication-protocol-v1-0-4/
4. https://www.dobot.cc/download/dobot-magician-api-v1-1-1/

5 Laboratory

Complete the exercises and save all code and follow the procedure to turn in your work.

5.1 Requirements

Bring your raspberry pi (RPI) so that you can develop your code Robot routine.

5.2 Procedure

1. Download the coding files and create your own header and implementation files for your class. Use the provided main file for your code.
2. You can test out your code using my example main function.
3. Save all work to be submitted in your final project.

5.3 Exercises

Develop a class that inherits the provided Robot class, I called mine Dobot. Within the class create a member of type ***std::function*** (see layout). Create a routine that can successfully pick up ping pong balls on the grate when their positions are known and place them into the rails. Use `goToXYZ` to move the robot to a XYZ coordinate. Use the `setPump` function to turn on and off the suction pump. Use the `wait` function to pause the routine for a certain time period. When finished, demonstrate your code to GTA. Use the `setup` function to setup the Dobot. Use the following command to compile.

```
sudo g++ -o lab6 lab6_main.cpp Message.h Message.cpp Packet.h Packet.cpp ProtocolDef.h
Protocol.h Protocol.cpp ProtocolID.h RingBuffer.h RingBuffer.cpp Robot.h Robot.cpp
Dobot.h command.h command.cpp -lwiringPi
```

Appendix

	ALT5	ALT4	ALT3	ALT2	ALT1	ALT0	WPi	Pull	Mode	Pin Numbers		Mode	Pull	WPi	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	
RPi A, B, A+, B+, 2/3	50mA maximum on 3.3V supply								3.3V	1	2	5V	max current draw ~300mA (cover when not in use)								
	(note 2)			reserved	SA3	I2C1 SDA	8	up	GPIO2	3	4	5V	max current draw ~300mA (cover when not in use)								
	(note 2)			reserved	SA2	I2C1 SCL	9	up	GPIO3	5	6	GND									
	ARM_TDI			reserved	SA1	GPCLK0	7	up	GPIO4	7	8	GPIO14	down	15	TXD0	SD6				TXD1	
									GND	9	10	GPIO15	down	16	RXD0	SD7				RXD1	
	RTS1	SPI1_CE1_N	RTS0	reserved	SD9	reserved	0	down	GPIO17	11	12	GPIO18	down	1	PCM_CLK	SD10		BSCSL_SDA/MOSI	SPI1_CE0_N	PWM0	
		ARM_TMS	SD1_DAT3	reserved	reserved	reserved	2	down	GPIO27	13	14	GND									
		ARM_TRST	SD1_CLK	reserved	SD14	reserved	3	down	GPIO22	15	16	GPIO23	down	4	reserved	SD15	reserved	SD1_CMD	ARM_RTCK		
	50mA maximum on 3.3V supply								3.3V	17	18	GPIO24	down	5	reserved	SD16	reserved	SD1_DAT0	ARM_TDO		
				reserved	SD2	SPI0_MOSI	12	down	GPIO10	19	20	GND									
				reserved	SD1	SPI0_MISO	13	down	GPIO9	21	22	GPIO25	down	6	reserved	SD17	reserved	SD1_DAT1	ARM_TCK		
				reserved	SD3	SPI0_CLK	14	down	GPIO11	23	24	GPIO8	up	10	SPI_CE0_N	SD0	reserved				
									GND	25	26	GPIO7	up	11	SPI_CE1_N	SWE_N / SRW_N	reserved				
RPi A+, B+, 2/3 Only	Do not use (GPIO0) -- see note 3			reserved	SA5	SDA0	30	up	ID_SD	27	28	ID_SC	up	31	SCL0	SA4	reserved	Do not use (GPIO1) -- see note 3			
	ARM_TDO			reserved	SA0	GPCLK1	21	up	GPIO5	29	30	GND									
	ARM_RTCK			reserved	SOE_N/SE	GPCLK2	22	up	GPIO6	31	32	GPIO12	down	26	PWM0	SD4	reserved			ARM_TMS	
	ARM_TCK			reserved	SD5	PWM1	23	down	GPIO13	33	34	GND									
				reserved	SD11	PCM_FS	24	down	GPIO19	35	36	GPIO16	down	27	reserved	SD8	reserved	CTS0	SPI1_CE2_N	CTS1	
		ARM_TDI	SD1_DAT2	reserved	reserved	reserved	25	down	GPIO26	37	38	GPIO20	down	28	PCM_DIN	SD12	reserved	BSCSL / MISO	SPI1_MOSI	GPCLK0	
									GND	39	40	GPIO21	down	29	PCM_DOUT	SD13	reserved	BSCSL / CE_N	SPI1_SCLK	GPCLK1	

Type	Linux DT	Description
GPIO	sysfs	general purpose input/output
SPI	spi	serial peripheral interface
I2C	i2c0/i2c1	I^2C Bus
UART	uart0	UART
PWM	pwm	Pulse Width Modulation
GPCLK	gp_clk	General purpose clock (GPCLK1 is reserved)
PCM	pcm	PCM audio
SA	smi	Secondary Memory Interface

Note 1: The data in this table was created from the www.eLinux.org web pages, system information, and datasheets where available.

Note 2: On early models of the RPi, Pin 3 is GPIO0 and Pin 5 is GPIO1. Also, these pins have permanent on-board 1.8 KΩ pull-up resistors attached (for the I^2C bus).

Note 3: ID_SD and ID_SC pins are reserved for the ID EEPROM (for different HATs). This is an I^2C interface that is probed at boot time in order to detect attached boards. This allows Linux to load the correct drivers for a HAT. See Chapter 8.

Register	Offset	MSB	32-bits for each register	LSB

Offset is from the virtual address 0x3F000000 on the RPi 2/3, and 0x20000000 on all other RPi models.

GPFSELn	The Function Select mode for each GPIO (3 bits for each GPIO and 54 GPIOs in total)											Read/Write
Bits		[31-30]	[29-27]	[26-24]	[23-21]	[20-18]	[17-15]	[14-12]	[11-9]	[8-6]	[5-3]	[2-0]
GPFSEL0	0000	X	FSEL9	FSEL8	FSEL7	FSEL6	FSEL5	FSEL4	FSEL3	FSEL2	FSEL1	FSEL0
GPFSEL1	0004	X	FSEL19	FSEL18	FSEL17	FSEL16	FSEL15	FSEL14	FSEL13	FSEL12	FSEL11	FSEL10
GPFSEL2	0008	X	FSEL29	FSEL28	FSEL27	FSEL26	FSEL25	FSEL24	FSEL23	FSEL22	FSEL21	FSEL20
GPFSEL3	000C	X	FSEL39	FSEL38	FSEL37	FSEL36	FSEL35	FSEL34	FSEL33	FSEL32	FSEL31	FSEL30
GPFSEL4	0010	X	FSEL49	FSEL48	FSEL47	FSEL46	FSEL45	FSEL54	FSEL43	FSEL42	FSEL41	FSEL40
GPFSEL5	0014	[32-12] X							FSEL53	FSEL52	FSEL51	FSEL50

GPSETn	The Output Set register - use this to set a GPIO high (1 bit for each of the 54 GPIOs)	Read/Write
GPSET0	001C	[31-0] mapped to GPIO31 to GPIO0 (1 = set GPIO)(0 = no effect)
GPSET1	0020	[31-22] X / [21-0] mapped to GPIO53 to GPIO32 (1 = set GPIO)(0 = no effect)

GPCLRn	The Output Clear register - use this to set a GPIO low (1 bit for each of the 54 GPIOs)	Read/Write
GPCLR0	0028	[31-0] mapped to GPIO31 to GPIO0 (1 = clear GPIO)(0 = no effect)
GPCLR1	002C	[31-22] X / [21-0] mapped to GPIO53 to GPIO32 (1 = clear GPIO)(0 = no effect)

GPLVLn	The Level Read register - use this to read the value of a GPIO (1 bit for each of the 54 GPIOs)	Read Only
GPLVL0	0034	[31-0] mapped to GPIO31 to GPIO0 (1 = level is high)(0 = level is low)
GPLVL1	0038	[31-22] X / [21-0] mapped to GPIO53 to GPIO32 (1 = level is high)(0 = level is low)

GPPUD	The Pull-up/Pull-down Enable register - use this to define the configuration	Read/Write
GPPUD	0094	[31-2] X / [1-0]

GPPUDCLKn	The Pull-up/Pull-down Enable Clock register - use this to apply GPPUD to a particular GPIO	Read/Write
GPPUDCLK0	0098	[31-0] mapped to GPIO31 to GPIO0 (1 = assert clock)(0 = no effect)
GPPUDCLK1	009C	[31-22] X / [21-0] mapped to GPIO53 to GPIO32 (1 = assert clock)(0 = no effect)

Function	GPFSELn
Bits	Meaning
000	input
001	output
100	ALT0
101	ALT1
110	ALT2
111	ALT3
011	ALT4
010	ALT5

Example: Set GPIO17 to be an output and set it high.
Solution: Write bits 001 to FSEL17, which is bits 21, 22, and 23 of the GPFSEL1 register to set the pin up as an output.
Then write a 1 to bit 17 of the GPSET0 register to set the output high.

Function	PUD
Bits	Meaning
00	no pull-up/down
01	pull-down
10	pull-up
11	X

FSEL17

GPFSEL1

[31]	[30]	...	[24]	[23]	[22]	[21]	[20]	...	[0]
X	X	...	?	0	0	1	?	...	?

Do not change the "?" bits as this will affect other GPIO select modes!

Figure 3 – Raspberry Pi 2/3 GPIO memory addressing information

Name	Function	See section
SDA0	BSC[5] master 0 data line	BSC
SCL0	BSC master 0 clock line	BSC
SDA1	BSC master 1 data line	BSC
SCL1	BSC master 1 clock line	BSC
GPCLK0	General purpose Clock 0	<TBD>
GPCLK1	General purpose Clock 1	<TBD>
GPCLK2	General purpose Clock 2	<TBD>
SPI0_CE1_N	SPI0 Chip select 1	SPI
SPI0_CE0_N	SPI0 Chip select 0	SPI
SPI0_MISO	SPI0 MISO	SPI
SPI0_MOSI	SPI0 MOSI	SPI
SPI0_SCLK	SPI0 Serial clock	SPI
PWMx	Pulse Width Modulator 0..1	Pulse Width Modulator
TXD0	UART 0 Transmit Data	UART
RXD0	UART 0 Receive Data	UART
CTS0	UART 0 Clear To Send	UART
RTS0	UART 0 Request To Send	UART
PCM_CLK	PCM clock	PCM Audio
PCM_FS	PCM Frame Sync	PCM Audio
PCM_DIN	PCM Data in	PCM Audio
PCM_DOUT	PCM data out	PCM Audio
SAx	Secondary mem Address bus	Secondary Memory Interface
SOE_N / SE	Secondary mem. Controls	Secondary Memory Interface
SWE_N / SRW_N	Secondary mem. Controls	Secondary Memory Interface
SDx	Secondary mem. data bus	Secondary Memory Interface
BSCSL SDA / MOSI	BSC slave Data, SPI salve MOSI	BSC ISP slave
BSCSL SCL / SCLK	BSC slave Clock, SPI slave clock	BSC ISP slave
BSCSL - / MISO	BSC <not used>,SPI MISO	BSC ISP slave
BSCSL - / CE_N	BSC <not used>, SPI CSn	BSC ISP slave

Name	Function	See section
SPI1_CEx_N	SPI1 Chip select 0-2	Auxiliary I/O
SPI1_MISO	SPI1 MISO	Auxiliary I/O
SPI1_MOSI	SPI1 MOSI	Auxiliary I/O
SPI1_SCLK	SPI1 Serial clock	Auxiliary I/O
TXD0	UART 1 Transmit Data	Auxiliary I/O
RXD0	UART 1 Receive Data	Auxiliary I/O
CTS0	UART 1 Clear To Send	Auxiliary I/O
RTS0	UART 1 Request To Send	Auxiliary I/O
SPI2_CEx_N	SPI2 Chip select 0-2	Auxiliary I/O
SPI2_MISO	SPI2 MISO	Auxiliary I/O
SPI2_MOSI	SPI2 MOSI	Auxiliary I/O
SPI2_SCLK	SPI2 Serial clock	Auxiliary I/O
ARM_TRST	ARM JTAG reset	<TBD>
ARM_RTCK	ARM JTAG return clock	<TBD>
ARM_TDO	ARM JTAG Data out	<TBD>
ARM_TCK	ARM JTAG Clock	<TBD>
ARM_TDI	ARM JTAG Data in	<TBD>
ARM_TMS	ARM JTAG Mode select	<TBD>

Figure 4 – BCM2835 peripherals function descriptions