EE405 Electronic Design Lab – Robotic Manipulator

Lab 2. Switch Circuit Control

- Control the solenoid magnet(end effector) via GPIO with C++ code

TA : Hyojun Lee
junelhdove@kaist.ac.kr

Purpose of the session

Control the solenoid magnet(end effector) via GPIO with C++ code

- Configuring a circuit
  - Building a simple **switching circuit*** for safe GPIO control
- Control GPIO
  - Learning how to access hardware from software (**linux kernel**)
  - How to control GPIO using code (**shell script/C++)**

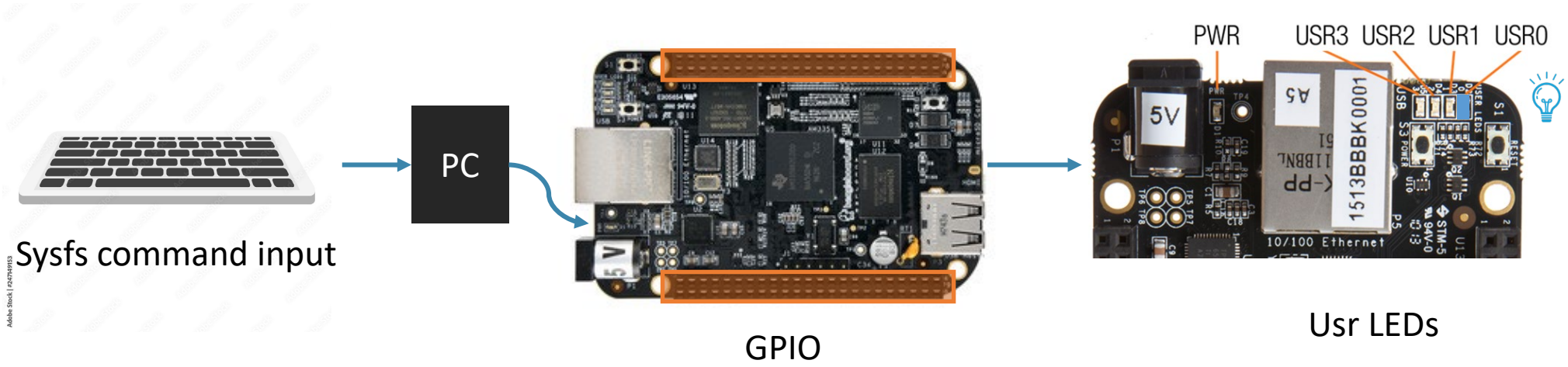
➡ Control the solenoid magnet(end effector) with C++ code

***switching circuit :** wired circuit which transistor used for electric switch operation

Contents

Lab 2 – Prob 2A. **User LED0 Control using sysfs and command line**

- **Sysfs / GPIO**
- **GPIO for USR LEDs**



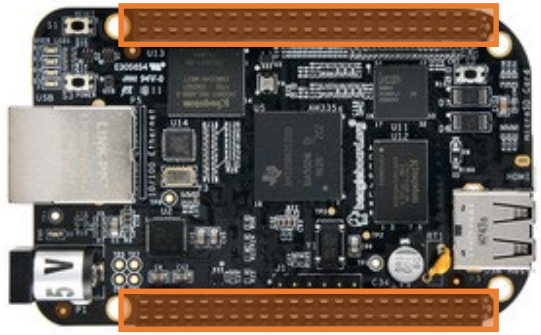
Prob 2A. - Background information

GPIO / sysfs

- **Sysfs(Sys FileSystem)**
  - Filesystem for exporting kernel objects(??)
  - Filesystem which provides information about devices, filesystems, components.

  For the beaglebone board, GPIO information is accessible using sysfs.
- **GPIO(General-Purpose Input-Output)**
  - Digital pin on the Beaglebone board works as GPIO.
  - We will access / modify GPIO values through sysfs, with simple command(bash script, C++ code).

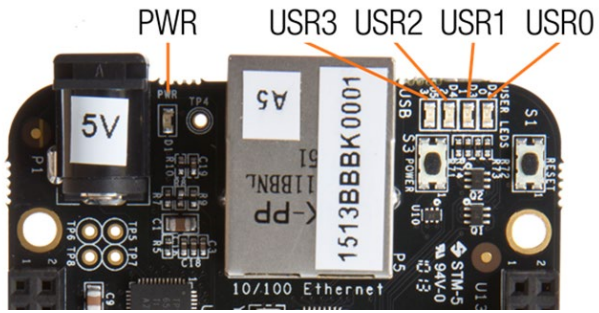
GPIO Pins(in Orange)



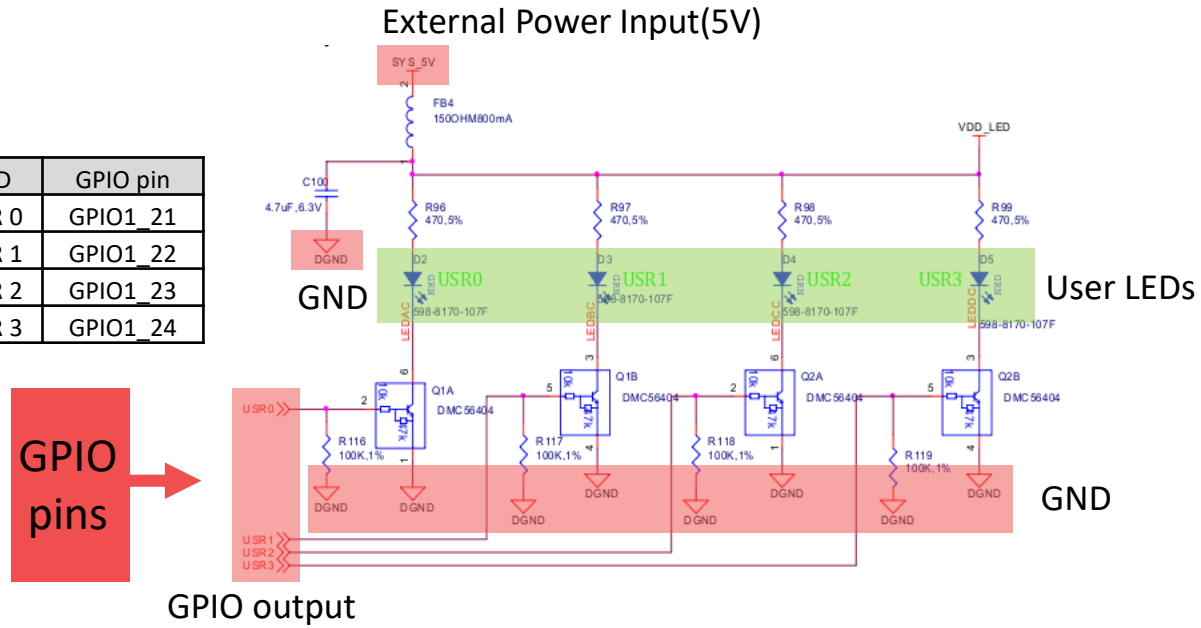
Prob 2A. - Background information

USR LEDs on Beaglebone Board

- **USR LEDs on BeagleBone**



LED	GPIO pin
USR 0	GPIO1_21
USR 1	GPIO1_22
USR 2	GPIO1_23
USR 3	GPIO1_24



Prob 2A. - Contents

GPIO control command

- **Example command**
  - If you want to illuminate USR0 LED,

```
# cd /sys/class/leds
# cd beaglebone\:green\:usr0                                    // access USR0 LED directory
# ls -F
brightness  invert           power/      trigger
device@     max_brightness   subsystem@  uevent
# echo none > trigger
# echo 1 > brightness
# echo 0 > brightness                                  // changes the element for USR0 LED
```

Sysfs directory to the USR LEDs

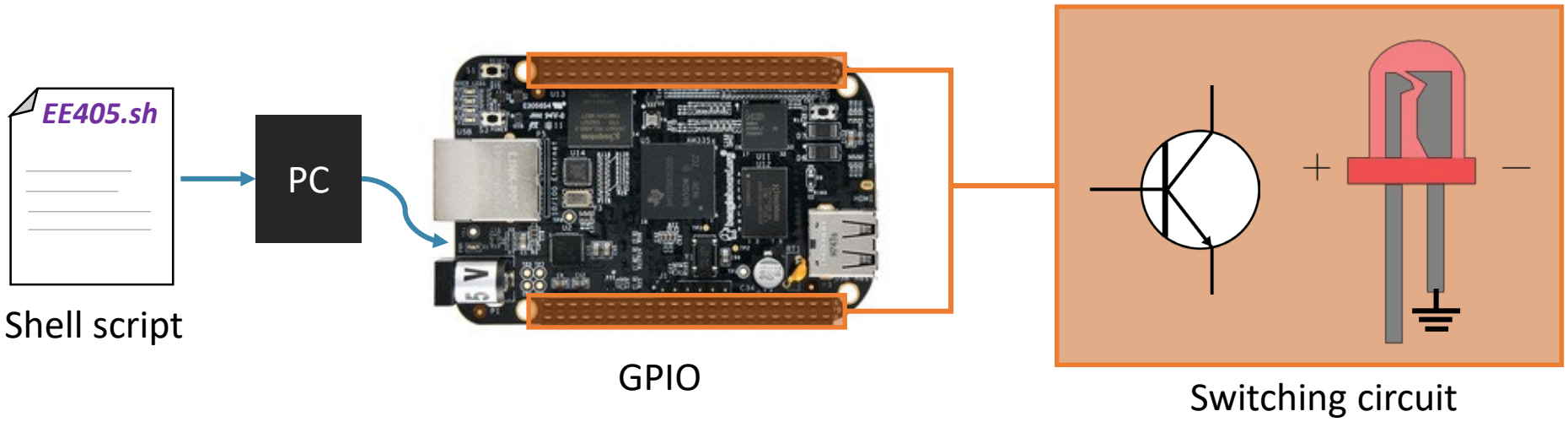
`/sys/class/leds`

Sysfs directory to the GPIO ports

`/sys/class/gpio`

Lab 2 – Prob 2B. LED Lights Control with shell script

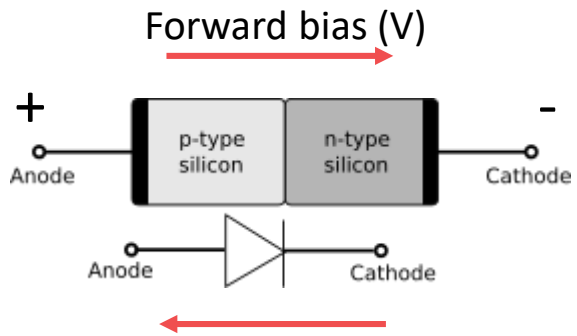
- **Electric parts**
  - **Diode, LED, Transistor**
  - **How to wire circiut guarantees safety?**
- **Shell scripting**



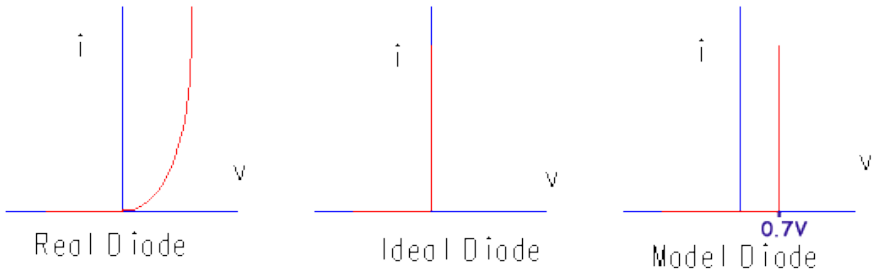
Prob 2B. - Background information

Electric parts

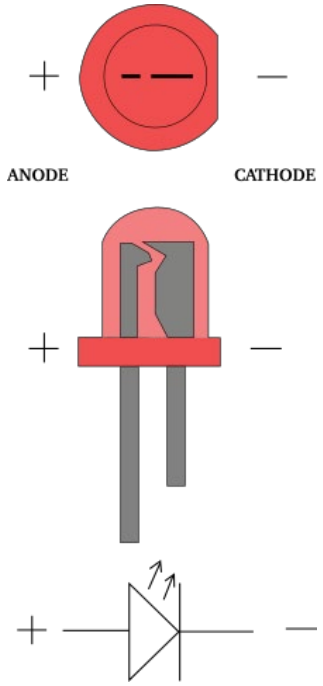
- **LED(Light Emitting Diode)**
  - Diode



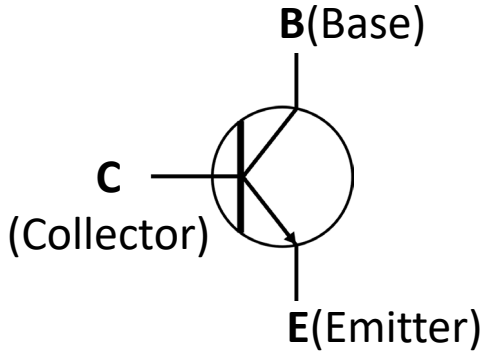
Backward bias (little or no current)



  - LED



  - Transistor



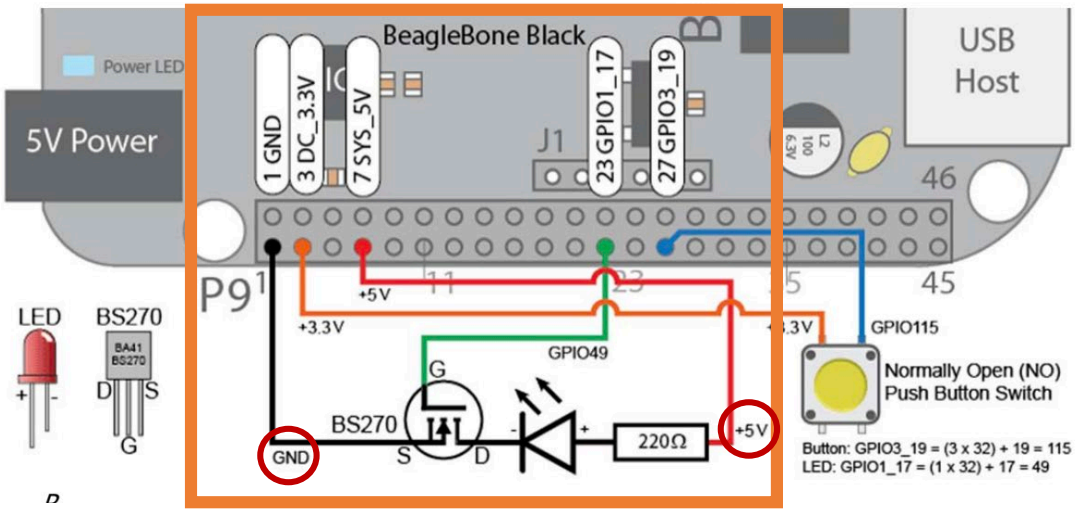
uLN2803AN(transistor array)

Prob 2B. - Background information

Electric parts

- **Switching circuit example)**
  - circuit connection:

    GPIO Signal -> Collector
    +5V -> Base / Emitter -> GND

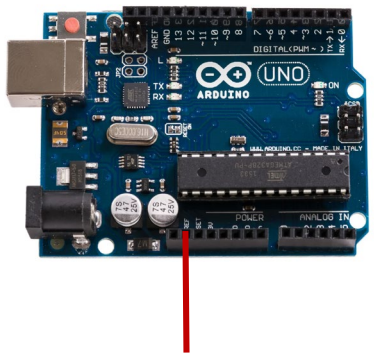



```
# cd /sys/class/gpio
# echo 30 > export          // export GPIO port
# cd gpio30
# echo out > direction  // set GPIO direction
# echo 1 > value
# echo 0 > value                 // turn ON/OFF
# cd /sys/class/gpio
# echo 30 > unexport     // unexport GPIO port
```

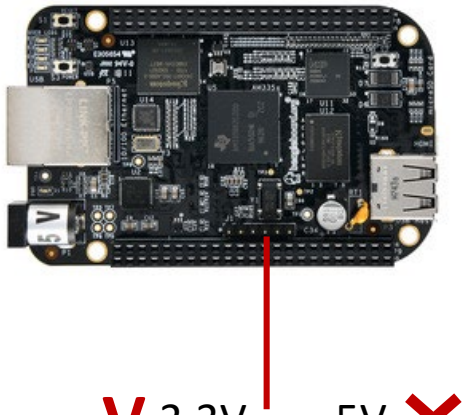
Prob 2B. - Background information

Electric parts

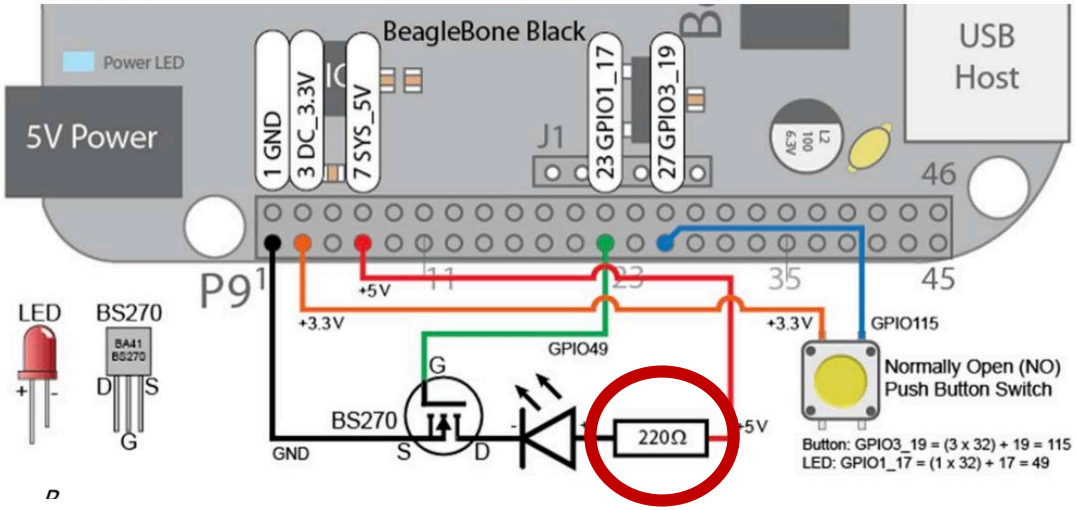
- **Resistor**
  - Is resistor connection essential?
  - Direct connection to external power:



V 3.3V V 5V



V 3.3V 5V ✕

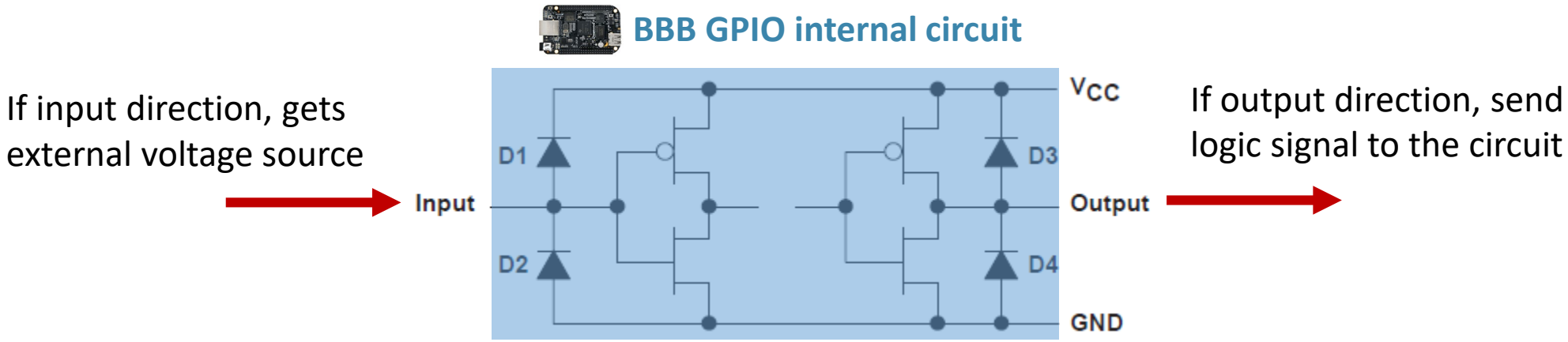


Burns the BBB!

Prob 2B. - Background information

Safe electric circuit connection

- **How to connect safely?**
  - Should limit current inside the Beaglebone lower than **1mA**(Recommended current limitation)
    (maximum current allowed for BBB : 8mA)

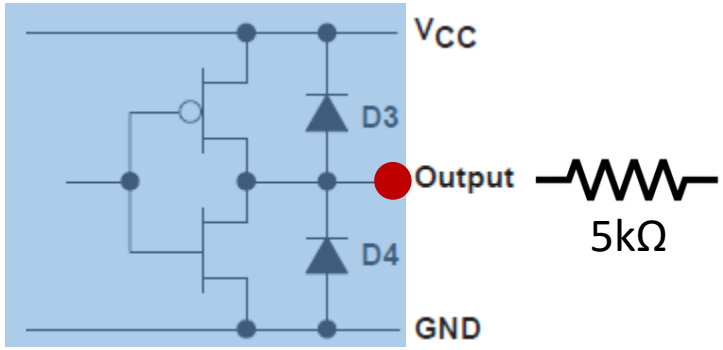


Prob 2B. - Background information

Safe electric circuit connection

- **How to connect safely?**
  - Should limit current inside the Beaglebone lower than 1mA(Recommended current limitation)
    (maximum current allowed for BBB : 8mA)
  - GPIO output direction case

GPIO, **Direction = output**



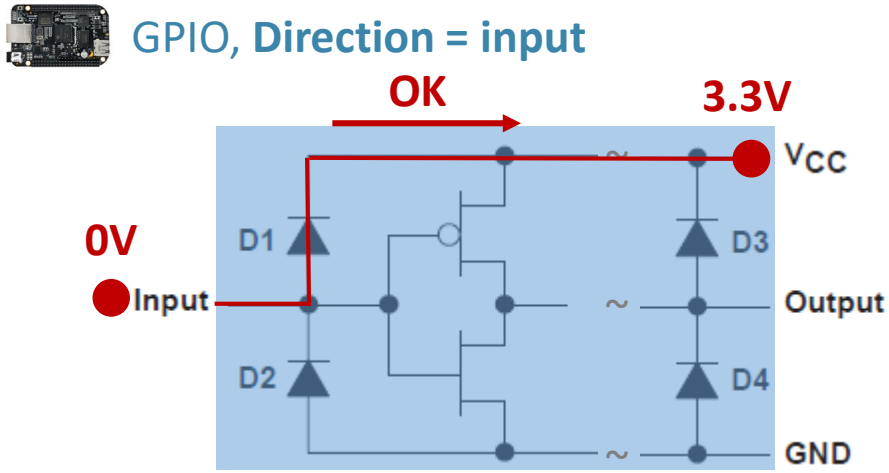
GPIO value	Output voltage
0	0V
1	3.3V(=Vcc)

Maximum current = $\frac{3.3\text{V}-0\text{V}}{5\text{k}\Omega} = 0.66\text{mA}$
(If GPIO output connected to 0V)

Prob 2B. - Background information

Safe electric circuit connection

- **How to connect safely?**
  - Should limit current inside the Beaglebone lower than 1mA(Recommended current limitation)
    (maximum current allowed for BBB : 8mA)
  - GPIO input direction case



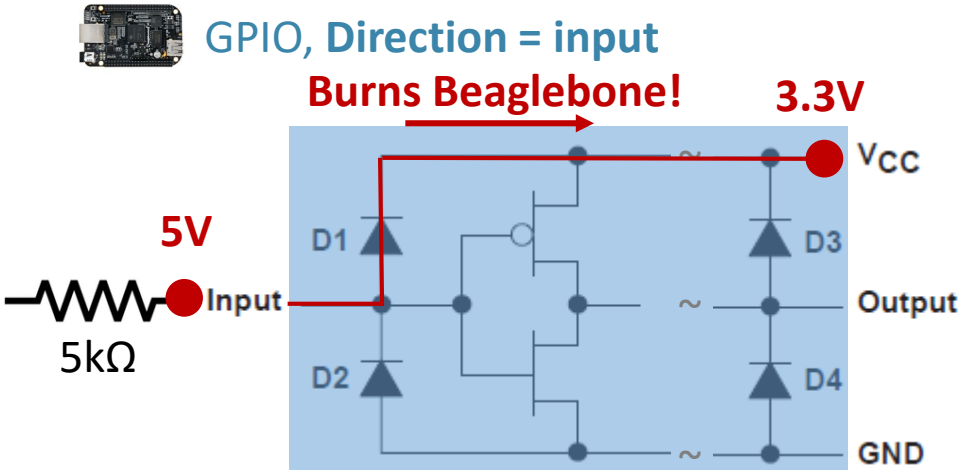
External voltage source	Input Voltage	Vcc	Current (Input-Vcc)
0V	0V	3.3V	~= 0
5V	5V	3.3V	Inf

No problem occurs.

Prob 2B. - Background information

Safe electric circuit connection

- **How to connect safely?**
  - Should limit current inside the Beaglebone lower than 1mA(Recommended current limitation)
    (maximum current allowed for BBB : 8mA)
  - GPIO input direction case



External voltage source	Input Voltage	Vcc	Current (Input-Vcc)
0V	0V	3.3V	~= 0
5V	5V	3.3V	Inf

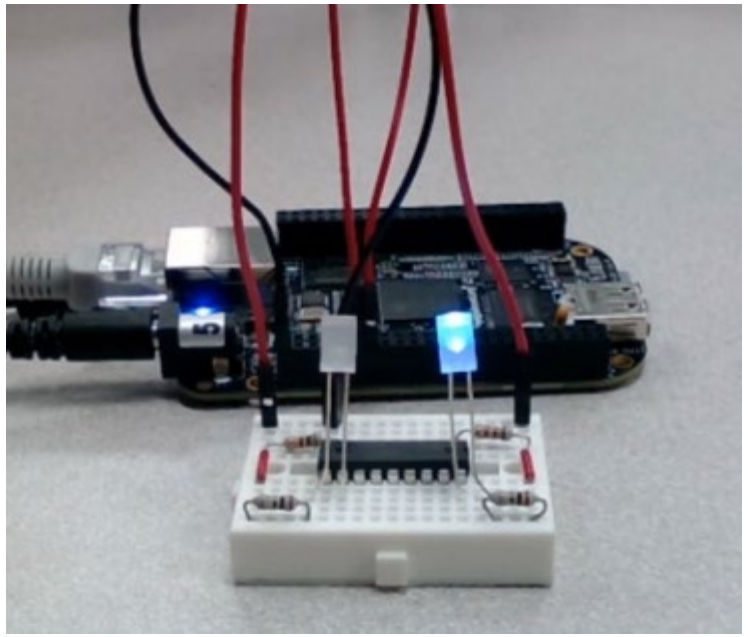
VERY HIGH current flows -> Burns beaglebone!

With 5kΩ resistor, current = $\frac{5\text{V}-3.3\text{V}-0.7\text{V}}{5\text{k}\Omega} = 0.2\text{mA}$

Prob 2B. - Contents

LED lights control with shell script

- Switching circuit design
  - Wire electric circuit to the GPIO with LEDs, transistor, resistor.
  - For the connection sequence, please refer to the lab experimental guide.



Prob 2B. - Contents

LED lights control with shell script

- ui_control_light.sh
  - Test light control along with user input.

```
#!/bin/bash

# ui_control_lights.sh
# This code tests for the time takes for LED light control for many loops.

echo "ui_control_lights.sh"

echo "Export: Get access permission for GPIO30 & 31"
echo 30 > /sys/class/gpio/export
echo 31 > /sys/class/gpio/export

echo "Set directions of GPIO 30 & 31 as output"
echo out > /sys/class/gpio/gpio30/direction
echo out > /sys/class/gpio/gpio31/direction

while true
do

read -p "Type 'light_id' and 'on_off'" lid on_off

if [ $lid -lt 1 ]
then
echo "Input light_id is less than 1: Exit"
break
fi

if [ $lid -gt 2 ]
then
echo "Error Light_ID: 0:Exit, 1:Light_1, 2:Light_2"
continue
fi

if [ $on_off != "on" -a $on_off != "off" ]
then
echo "Error ON_OFF: Please type 'on' or 'off'"
continue
fi

if [ $lid -eq 1 ]
then
    if [ $on_off == "on" ]
    then
        echo 1 > /sys/class/gpio/gpio30/value
    fi
    if [ $on_off == "off" ]
    then
```

- Full code will be given.
- Understand the code by wrting by your own.
- Execute it, and test your circuit.

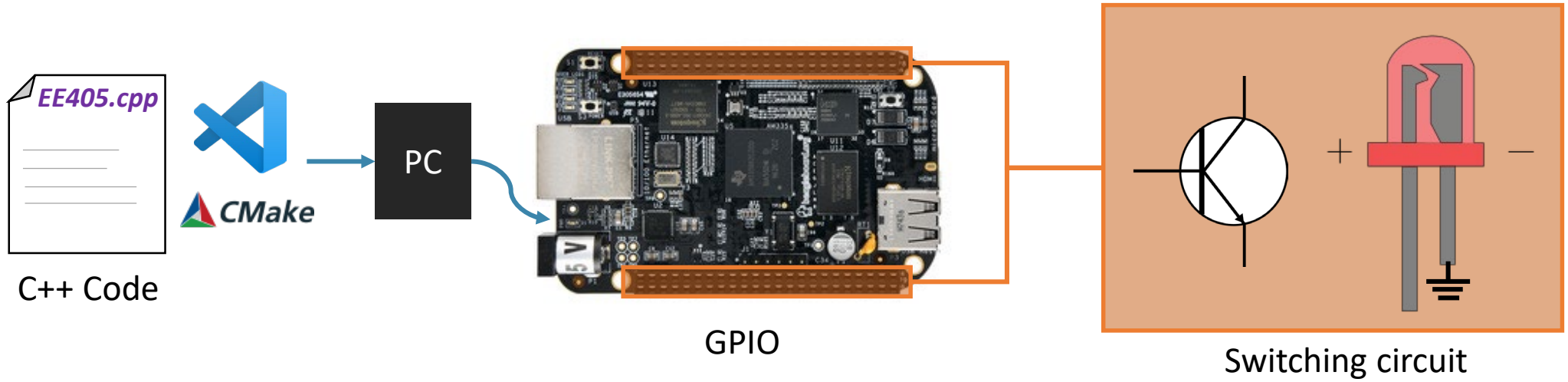
- loop_control_lights.sh
  - Test time consumption of the ON/OFF light control loop for N times.

```
#!/bin/bash

# loop_control_lights.sh
# This code tests for the time takes for LED light control for many loops.

echo "loop_control_lights.sh"

# ------------------------------------------

# 1) set GPIO 30/31 as export
# !! REPLACE THIS PART TO YOUR CODE !!

# ------------------------------------------

# ------------------------------------------

# 2) set GPIO 30/31 direction as output
# !! REPLACE THIS PART TO YOUR CODE !!

# ------------------------------------------

# 3) save the start time(before the light control)
start=$(date +%s.%N)

for ((i=0;i<10;i++))
do

  # ------------------------------------------

  # 4) LED Light control loop - Please refer to the experiment guide
  # !! REPLACE THIS PART TO YOUR CODE !!

  # ------------------------------------------

done

# 5) save the end time(after the light control)
end=$(date +%s.%N)
echo "Start:$start"I End:$end"

# ------------------------------------------

# 6) set GPIO 30/31 direction as input
# !! REPLACE THIS PART TO YOUR CODE !!

# ------------------------------------------
```

- Skeleton code will be given.
- Complete the code, execute it.
- **Record the time consumption** for your lab report.

Lab 2 – Prob 2C. LED Lights Control using C++ / Sysfs

- **C++ code to control GPIO**



Prob 2C. - Contents

LED lights control with C++ executables

- GPIO control through C++ executables, using same electric circuit.
  - How to control via C++? You can check basic idea on:
    - Access GPIO from Linux user space

    [http://falsinsoft.blogspot.kr/2012/11/access-gpio-from-linux-user-space.html]
    - How to use GPIO signals in Linux, C/C++

    [https://www.ics.com/blog/how-control-gpio-hardware-c-or-c]

  - Generate header file / source code for GPIO control functions
  - Using the functions defined, complete the code to test the GPIO.

  - Skeleton codes will be given.

Prob 2C. - Contents

LED lights control with C++ executables

- gpio_control.hpp [Given]
  - Header file for GPIO control functions

```
// File gpio_control.hpp
// Function definitions for gpio_control.cpp

#define MAX_BUF 64 /* For the max length of string */

int gpio_export(unsigned int gpio); // gpio means gpio number (0 to 127)
/* - input : GPIO port number to export
  - function that exports GPIO port */

int gpio_unexport(unsigned int gpio);
/* - input : GPIO port number to unexport
  - function that unexports GPIO port */

int gpio_set_dir(unsigned int gpio, unsigned int out); // out = 0: in. out = 1: out.
/* - input : GPIO port number to set direction / desired direction(out : 1, in : 0)
  - function sets the direction of the exported GPIO port */

int gpio_fd_open(unsigned int gpio); //Returns file descriptor fd
/* - input : GPIO port number to get fd
  - output : fd(file descriptor; information used to write values to the configured GPIO)
  - function which opens the GPIO port and get the file descriptor info */

int gpio_fd_close(int fd);
/* - function which closes the GPIO port using file descriptor information */

int gpio_fd_set_value(int fd, unsigned int value); // value can be 0 or 1
/* - input : GPIO port file descriptor / desired value(1: ON, 0: OFF)
  - Sample code to turn on/off the GPIO (IN CASE OF out direction) */
```

- gpio_control.cpp
  - body of GPIO control functions

```
#include "../include/gpio_control.hpp"

#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>

int gpio_export(unsigned int gpio)
{

/*

  - input : GPIO port number to export
  - function that exports GPIO port

  !! MODIFY FOLLOWING SAMPLE CODE PROPERLY !!

  int fd;
  char buf[MAX_BUF];
  int gpio = XX;
  fd = open("/sys/class/gpio/export", O_WRONLY);
  sprintf(buf, "%d", gpio);
  write(fd, buf, strlen(buf));
  close(fd);

*/

}

int gpio_unexport(unsigned int gpio)
{

/*

  - input : GPIO port number to unexport
  - function that unexports GPIO port

  !! MODIFY FOLLOWING SAMPLE CODE PROPERLY !!

  fd = open("/sys/class/gpio/unexport", O_WRONLY);
  sprintf(buf, "%d", gpio);
  write(fd, buf, strlen(buf));
  close(fd);

*/

}
```

- Reference code will be given.
- Complete the code.

Prob 2C. - Contents

LED lights control with C++ executables

- test_light_control.cpp
  - Test light control along with user input.

```c
#include <stdio.h>
#include <string.h>
#include "../include/gpio_control.hpp"

int main(int argc, char *argv[])
{
  int light_id;
  int fd_gpio_30, fd_gpio_31;
  int on_off_int;
  char on_off_str[8];

  /* 0.Print title */
  printf("test_light_control\n");

  /* 1. Set variables: light_id = 1; */
  light_id = 1;

/*
  2. Export GPIO 30 & GPIO31

  !! REPLACE THIS PART TO YOUR CODE !!

*/

/*
  3. Set direction of GPIO 30 & GPIO31 to out.

  !! REPLACE THIS PART TO YOUR CODE !!

*/

/*
  4. Open GPIO30 & GPIO31.

  !! REPLACE THIS PART TO YOUR CODE !!

*/

  /* 5. Loop while light_id > 0 */
  while(light_id > 0)
  {
    /* A. Prompt output */
    printf("Enter light_id and on_off_str: ");
    /* B. Get user input of light_id and on_off_str */
    scanf("%d %7s", &light_id, on_off_str);
    /* C. Check light_id. Break if <= 0. */
    if(light_id <= 0)
```

- Complete with the function you've generated.
- Execute it.

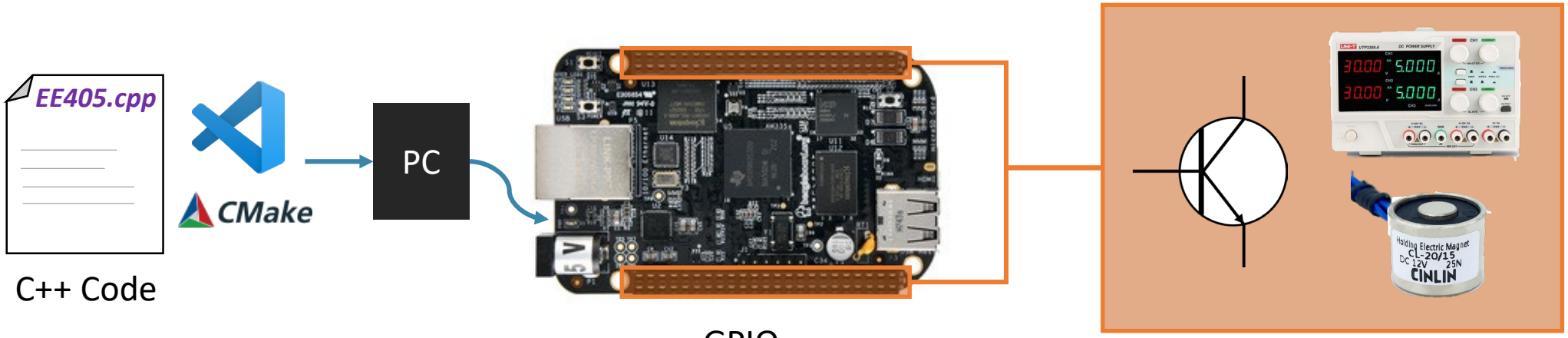
- loop_light_control.cpp
  - Test time consumption of the ON/OFF light control loop for N times.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
#include "../include/gpio_control.hpp"

int main(int argc, char *argv[])
{
  int fd_gpio_30, fd_gpio_31;

  struct timeval s_time, e_time;
  struct timezone tz;
  double start_time, end_time;

  /* 0.Print title */
  printf("loop_light_control\n");

/*
  1. Export GPIO 30 & GPIO31

  !! REPLACE THIS PART TO YOUR CODE !!

*/

/*
  2. Set direction of GPIO 30 & GPIO31 to out.

  !! REPLACE THIS PART TO YOUR CODE !!

*/

/*
  3. Open GPIO30 & GPIO31.

  !! REPLACE THIS PART TO YOUR CODE !!

*/

// 4. Save the start time(before the light control loop)
  gettimeofday(&s_time,&tz);

/*
  5. LED Control loop for 10 times

  !! REPLACE THIS PART TO YOUR CODE !!

*/
```

- Complete with the function you've generated.
- Execute it.
- **Record the time consumption** for your lab report.

Lab 2 – Prob 2D. Solenoid magnet control using C++ / Sysfs

- **Common ground in electric circuits**
- **C++ code to control GPIO**



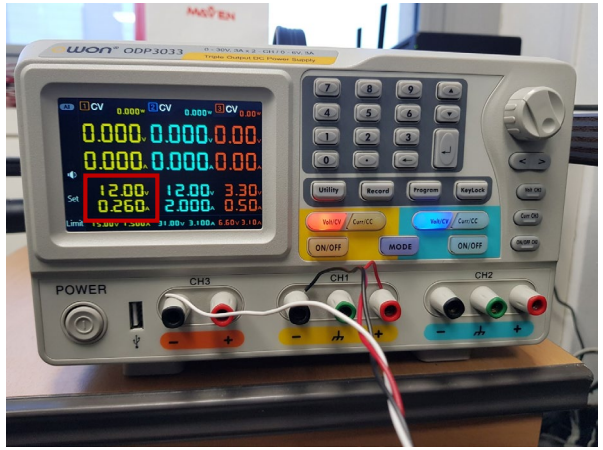
Prob 2D. – Background information

Common ground in electric circuits

- **Common ground in electric circuits**
  - Unlike LED lights, we will use DC power supply to control solenoid magnet.

MODEL	Voltage	12V	24V
CL-P20/15	Current	0.26A	0.13A
	Power	3W	
	Holding Force	25N/2.5KG	
Insulation grade		B(103°C)	
Dielectric Strength		AC600V 50/60HZ 1MIN	

Solenoid magnet we will use
Voltage = 12V, Current = 0.26A

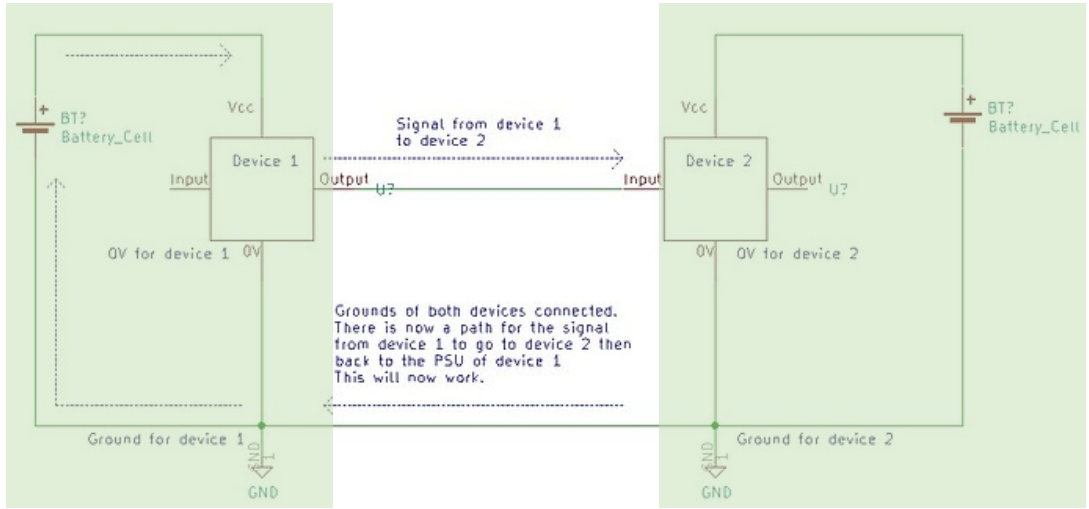


DC Power Supply
Voltage = 12V, Current = 0.26A

Prob 2D. - Background information

Common ground in electric circuits

- **Common ground in electric circuits**
  - If you have to connect 2 circuits from different power supply, common ground must be considered.
    = Connect different GNDs into one, to equalize voltage level



GND1 **GND2**

Prob 2D. - Contents

Solenoid magnet control using C++ / Sysfs

- Switching circuit for solenoid magnet
  - Wire electric circuit to the GPIO with solenoid magnet, transistor, resistor.
  - For the connection sequence, please refer to the lab experimental guide.
  - Please be aware to the commond ground!

Prob 2D. - Contents

Solenoid magnet control using C++ / Sysfs

- gripper_control.hpp
  - Header file for gripper control functions

```hpp
#ifndef GRIPPER_CONTROL_HPP
#define GRIPPER_CONTROL_HPP

// File gripper_control.hpp
// Function definitions for gripper_control.cpp

int gripper_open(unsigned int gpio);
// open the GPIO port for the solenoid magnet
// input : GPIO port number
// output : file descriptor of GPIO port

int gripper_on(unsigned int fd_gpio);
// turn on the solenoid magnet
// input : GPIO port number

int gripper_off(unsigned int fd_gpio);
// turn off the solenoid magnet
// input : GPIO port number

int gripper_close(unsigned int gpio, unsigned int fd_gpio);
// close the GPIO port for the solenoid magnet
// input : GPIO port number, file descriptor for GPIO

#endif
```

- gripper _control.cpp
  - body of gripper control functions
  - You might use functions defined on gpio_control

```cpp
#include <stdio.h>
#include <string.h>
#include <iostream>
#include "../include/gpio_control.hpp"

int gripper_open(unsigned int gpio)
{
  /*
    open the GPIO port for the solenoid magnet
    - input : GPIO port number
    - output : file descriptor of GPIO port

    !! FILL WITH YOUR CODE !!
  */
}

int gripper_on(unsigned int fd_gpio)
{
  /*
    turn on the solenoid magnet
    - input : GPIO port number

    !! FILL WITH YOUR CODE !!
  */
}

int gripper_off(unsigned int fd_gpio)
{
  /*
    turn off the solenoid magnet
    - input : GPIO port number

    !! FILL WITH YOUR CODE !!
  */
}

int gripper_close(unsigned int gpio, unsigned int fd_gpio)
{
  /*
    close the GPIO port for the solenoid magnet
    - input : GPIO port number, file descriptor for GPIO

    !! FILL WITH YOUR CODE !!
  */
}
```

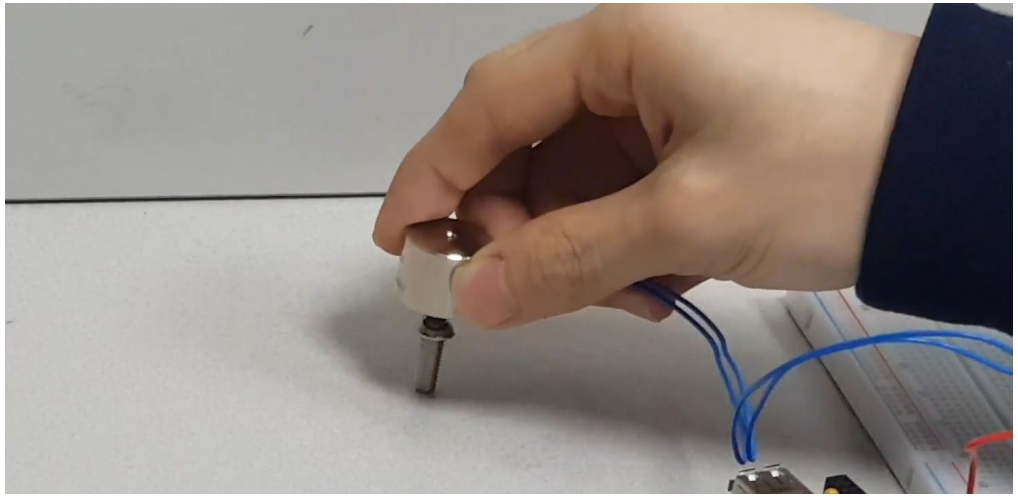
Prob 2D. - Contents

Solenoid magnet control using C++ / Sysfs

- gripper_test.cpp
  - Source code to test gripper control functions

```cpp
#include <stdio.h>
#include <string.h>
#include <chrono>
#include "../include/gpio_control.hpp"
#include "../include/gripper_control.hpp"

int main(int argc, char *argv[])
{

  /*
    1. Define variables for GPIO port number, GPIO file descriptor and length of time duration for holding an object
    !! FILL WITH YOUR CODE !!
  */

  printf("gripper_control\n");

  /*
    2. Open the gripper GPIO port
    !! FILL WITH YOUR CODE !!
  */

  /* Save start time of the loop */
  std::chrono::system_clock::time_point start_time = std::chrono::system_clock::now();
  std::chrono::system_clock::time_point current_time;
  std::chrono::microseconds loop_elasped_time_microsec;

  /* Turn on the gripper while pickup_time */
  while(loop_elasped_time_microsec.count()/1e6 < pickup_time)
  {
    current_time = std::chrono::system_clock::now();
    loop_elasped_time_microsec = std::chrono::duration_cast<std::chrono::microseconds>(current_time - start_time);
    /*
      3. Turn on the gripper
      !! FILL WITH YOUR CODE !!
    */
  }

  /*
    4. Close the gripper GPIO port
    !! FILL WITH YOUR CODE !!
  */

  return 0;
}
```

- Test it!



Lab Procedures

Switching circuit control

- First week(W4)
  - **Prob 2A.** User LED0 Control using sysfs and command line
    - Control GPIO through Sysfs command
  - **Prob 2B.** LED Lights Control with shell script
    - Wire switching circuit with LED lights
    - Control GPIO through shell script
- Second week(W5)
  - **Prob 2C.** LED Lights Control using C++ / Sysfs
    - Complete C++ source codes
    - Control GPIO through C++ executables
  - **Prob 2D.** Solenoid magnet control using C++ / Sysfs
    - Wire switching circuit with solenoid magnet
    - Complete C++ source codes
    - Control GPIO through C++ executables