2020F_ESE 3014_1

SEMESTER: 3rd

INSTRUCTOR: Linchen Wang

LAB: 7

DUE DATE: 26 Nov 2020 - 23:55

SUBMITTED DATE: 26 NOV 2020

STUDENTS: Amandeep Singh (C0765434)

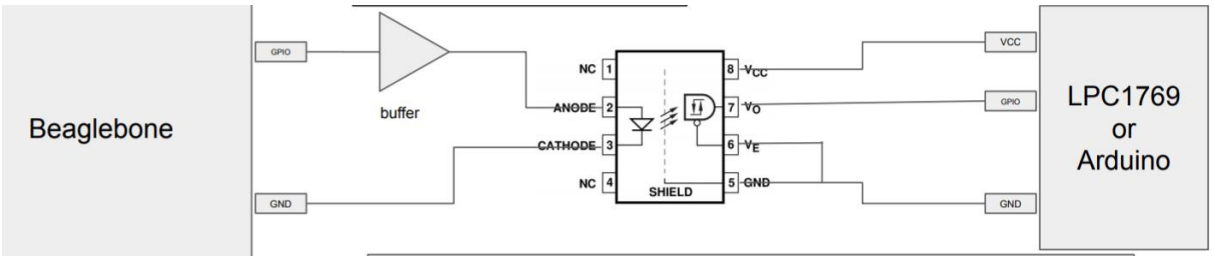
Lab-7

INTRODUCTION

In this Lab we are going to setup interfacing, with the BBB as master and Arduino as slave using GPIO pins.

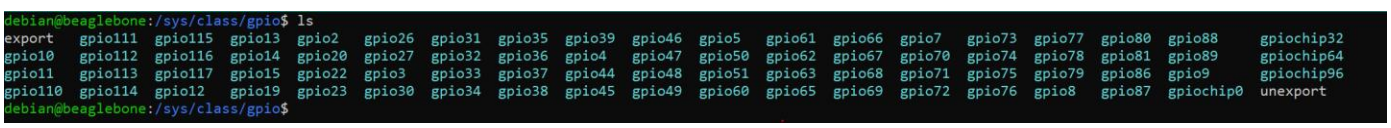
DESCRIPTION

- <u>Pin Configuration</u>:
  We use the following pin configuration for setting up communication between Arduino and BBB.



1. We can check the GPIO available to use by simply going to the following path and using the ls command

```
$cd /sys/class/gpio
$ls
```



```
debian@beaglebone:/sys/class/gpio$ ls
export   gpio111  gpio115  gpio13  gpio2   gpio26  gpio31  gpio35  gpio39  gpio46  gpio5   gpio61  gpio66  gpio7   gpio73  gpio77  gpio80  gpio88    gpiochip32
gpio10   gpio112  gpio116  gpio14  gpio20  gpio27  gpio32  gpio36  gpio4   gpio47  gpio50  gpio62  gpio67  gpio70  gpio74  gpio78  gpio81  gpio89    gpiochip64
gpio11   gpio113  gpio117  gpio15  gpio22  gpio3   gpio33  gpio37  gpio44  gpio48  gpio51  gpio63  gpio68  gpio71  gpio75  gpio79  gpio86  gpio9     gpiochip96
gpio110  gpio114  gpio12   gpio19  gpio23  gpio30  gpio34  gpio38  gpio45  gpio49  gpio60  gpio65  gpio69  gpio72  gpio76  gpio8   gpio87  gpiochip0 unexport
debian@beaglebone:/sys/class/gpio$
```

2. To export a GPIO we use the following command

```
$echo 60 > export
```

3. To check if the direction of the GPIO we use :

```
$cd gpio60
$cat direction
```

Note: to change the direction we use the command below

```
$ echo “out” > direction
```

```
debian@beaglebone:/sys/class/gpio$ cd gpio60
debian@beaglebone:/sys/class/gpio/gpio60$ cat direction
in
debian@beaglebone:/sys/class/gpio/gpio60$ echo "out" > direction
debian@beaglebone:/sys/class/gpio/gpio60$ cat direction
out
debian@beaglebone:/sys/class/gpio/gpio60$
```

4. To set these gpio high/low we use the following command:

```
$ echo 1 > value (on and gives 3.3v as output)
$ echo 0 > value (off and gives 0v as output)
```

5. Now we just have to run the code and make the connection and see if the setup works or not.

CONCLUSION

- Lab-7 was really important to understand how the GPIO interfacing of Arduino works with the beaglebone black and it can used in the future for our capstone project.
- GPIO are easy to work with and are bi directional.

APPENDIX

- Youtube link: **https://www.youtube.com/watch?v=Q4QisZvyhJo**

- GPIO.h CODE:

```
#define GPIO_PATH "/sys/class/gpio/"

namespace exploringBB {
  typedef int (*CallbackType)(int);
  enum GPIO_DIRECTION{ INPUT, OUTPUT };
  enum GPIO_VALUE{ LOW=0, HIGH=1 };
  enum GPIO_EDGE{ NONE, RISING, FALLING, BOTH };
  class GPIO {
  private:
    int number, debounceTime;
    string name, path;
  public:
    GPIO(int number); //constructor will export the pin
    virtual int getNumber() { return number; }
    // General Input and Output Settings
    virtual int setDirection(GPIO_DIRECTION);
    virtual GPIO_DIRECTION getDirection();
    virtual int setValue(GPIO_VALUE);
    virtual int toggleOutput();
    virtual GPIO_VALUE getValue();
    virtual int setActiveLow(bool isLow=true); //low=1, high=0
    virtual int setActiveHigh(); //default
    //software debounce input (ms) - default 0
    virtual void setDebounceTime(int time) { this->debounceTime = time; }
    // Advanced OUTPUT: Faster write by keeping the stream alive (~20X)
    virtual int streamOpen();
    virtual int streamWrite(GPIO_VALUE);
    virtual int streamClose();
    virtual int toggleOutput(int time); //threaded invert output every X ms.
    virtual int toggleOutput(int numberOfTimes, int time);
    virtual void changeToggleTime(int time) { this->togglePeriod = time; }
    virtual void toggleCancel() { this->threadRunning = false; }
    // Advanced INPUT: Detect input edges; threaded and non-threaded
    virtual int setEdgeType(GPIO_EDGE);
    virtual GPIO_EDGE getEdgeType();
    virtual int waitForEdge(); // waits until button is pressed
    virtual int waitForEdge(CallbackType callback); // threaded with callback
    virtual void waitForEdgeCancel() { this->threadRunning = false; }
    virtual ~GPIO(); //destructor will unexport the pin
    ...
     } /* namespace exploringBB */
```

Test.cpp code:

```
//Header files

#include<iostream>
#include<unistd.h> //for usleep
#include"GPIO.h"
using namespace exploringBB;
   using namespace std;
```

//MAINFUNCTION of Test.cpp

```
GPIO *out, *in; //global pointers
int activateLED(int var)
{
out->streamWrite(HIGH); //turn on the LED
return 0;
}

int main()
{
in= new GPIO(46); //pin to trigger the LED
out = new GPIO(60); //LED
in->setDirection(INPUT); //triggered enabled
out->setDirection(OUTPUT); //LED is an output
out->streamOpen(); //fast write to LED
out->streamWrite(LOW); //turn the LED off
in->setEdgeType(RISING); //wait for rising edge
in->waitForEdge(&activateLED); //pass the function
usleep(10000000); //allow 10 seconds
out->streamWrite(LOW); //turn off the LED after 10 seconds
out->streamClose(); //shutdown
return 0;
}
```

//code for Arduino

```
Void setup()
{
pinMode(2,INPUT); //reads from the GPIO
pinMode(4,OUTPUT);// used for the LED
}
Void loop()
{
Int val = digitalRead(2);
If (val==1)
      digitalWrite(4,HIGH);
else
      digitalWrite(4,LOW);

}
```