I2C Interfacing through Odroid-C2 and Arduino

Veera Shanti Ram†, Alok Ranjan Kesari† and G V V Sharma*

Contents

1 Sigle Slave

1.1 Configuring Arduino as slave

Problem 1.1. Run the following program on the odroid using Arduino software. This will configure the Arduino as a slave.

```
/**
 * Blink
 *
 * Turns on an LED on for one
    second,
 * then off for one second,
    repeatedly.
 */
#include "Arduino.h"
#include <Wire.h>

#define SLAVE_ADDRESS 0x04
int number = 0;
int state = 0;
void sendData();
void receiveData(int byteCount);

void setup() {
pinMode(13, OUTPUT);
Serial.begin(9600); // start
   serial for output
// initialize i2c as slave
Wire.begin(SLAVE_ADDRESS);

// define callbacks for i2c
   communication
Wire.onReceive(receiveData);
Wire.onRequest(sendData);

Serial.println("Ready!");
}

void loop() {
delay(100);
}

// callback for received data
void receiveData(int byteCount){

while(Wire.available()) {
number = Wire.read();
Serial.print("data received: ");
Serial.println(number);

if (number == 1){

if (state == 0){
digitalWrite(13, HIGH); // set the
    LED on
state = 1;
}
else{
digitalWrite(13, LOW); // set the
   LED off
state = 0;
}
}
}
}
```

† The authors were interns at IIT Hyderabad during the summer of 2017 email: veerashanthiram@gmail.com, alok.kesari@yahoo.co.in.
*The author is with the Department of Electrical Engineering, Indian Institute of Technology, Hyderabad 502285 India e-mail: gadepall@iith.ac.in.

```
// callback for sending data
void sendData(){
Wire.write(number);
}

//#ifndef LED_BUILTIN
//#define LED_BUILTIN 13
//#endif

//void setup()
//{
  //// initialize LED digital pin
     as an output.
  //pinMode(LED_BUILTIN, OUTPUT);
//}

//void loop()
//{
  //// turn the LED on (HIGH is
     the voltage level)
  //digitalWrite(13, 1);

  //// wait for a second
  //delay(500);

  //// turn the LED off by making
     the voltage LOW
  //digitalWrite(13,0);

   //// wait for a second
  //delay(500);
//}
```

1.2 On Odroid

The following commands are for an Odroid running Archlinuxarm. The process for running it on other Linux distributions is similar.

```
#For providing i2c interface
sudo pacman -S i2c-tools
#For smbus, python module to control i2c
sudo pip install cffi
sudo pip install smbus-cffi
#For starting the i2c module
sudo modprobe aml_i2c
#For checking if the slave is connected
sudo i2cdetect -y -r 1
```

The last command above will display the Arduino address as 0x04 on the terminal screen.

1.3 Connecting Odroid to Arduino

Problem 1.2. Connect the pins as in Table 1.

Odroid	Arduino
GND	GND
3	A4
5	A5

TABLE 1

1.4 Odroid as Master

Problem 1.3. Run the following code on Odroid and enter 1 when prompted. You will see the LED glow. Running it again will toggle the output.

```
import smbus
import time
# for RPI version 1, use "bus =
   smbus.SMBus(0)"
bus = smbus.SMBus(1)

# This is the address we setup in
   the Arduino Program
address = 0x04

def writeNumber(value):
        print("I am here")
        bus.write_byte(address,
           value)
# bus.write_byte_data(address, 0,
   value)
        return -1

def readNumber():
        number = bus.read_byte(
           address)
# number = bus.read_byte_data(
   address, 1)
        return number

while True:
        var = input("Enter 1-9:")
        if not var:
                continue
```

```
        writeNumber(var)
        print("RPI: Hi Arduino, I
            sent you ", var)
        # sleep one second
        time.sleep(1)

        number = readNumber()
        print("Arduino: Hey RPI, I
            received a digit ",
            number)
```

2 Multiple Slaves

Problem 2.1. Configure another arduino as a slave with address 0x05. Use problem 1.1.

Problem 2.2. Connect one more arduino to the odroid using a breadboard according to Table 1. You will have to make multiple connections using the breadboard.

Problem 2.3. Run the following command to check if both arduinos are detected.

```
sudo i2cdetect -y -r 1
```

Problem 2.4. Modify the code in problem 1.3 to verify if the i2c connection with the second arduino is active.

Problem 2.5. Run the following program to control both arduinos on the i2c bus.

```
import smbus
import time
# for RPI version 1, use "bus =
    smbus.SMBus(0)"
bus = smbus.SMBus(1)

# This is the address we setup in
    the Arduino Program
address1 = 0x04
address2 = 0x05

def writeNumber(address,value):
        bus.write_byte(address,
            value)
# bus.write_byte_data(address, 0,
    value)
        return -1

def readNumber(address):
        number = bus.read_byte(
            address)
# number = bus.read_byte_data(
    address, 1)
        return number

while True:
        var1 = input("Enter 1-9:")
        if not var1:
                continue

        writeNumber(address1,var1)
        print("RPI: Hi Arduino1, I
            sent you ", var1)
        # sleep one second
        time.sleep(1)

        var2 = input("Enter 1-9:")
        if not var2:
                continue
        writeNumber(address2,var2)
        print("RPI: Hi Arduino2, I
            sent you ", var2)
        # sleep one second
        time.sleep(1)

        number = readNumber(
            address1)
        print("Arduino1: Hey RPI,
            I received a digit ",
            number)
        number = readNumber(
            address2)
        print("Arduino2: Hey RPI,
            I received a digit ",
            number)
```

GPIO PIN-MAP

ODROID-C2 40pin Layout									Power Pin Special Function GPIO/Special Function	
WiringPi GPIO#	Export GPIO#	ODROID-C2 PIN	Label	HEADER		Label	ODROID-C2 PIN	Export GPIO#		WiringPi GPIO#
			3V3	1	2	5V0				
	205	I2CA_SDA	SDA1	3	4	5V0				
	206	I2CA_SCL	SCL1	5	6	GND				
7	249	GPIOX.BIT21	#249	7	8	TXD1	TXD_B	113		
			GND	9	10	RXD1	RXD_B	114		
0	247	GPIOX.BIT19	#247	11	12	#238	GPIOY.BIT10	238		1
2	239	GPIOX.BIT11	#239	13	14	GND				
3	237	GPIOX.BIT9	#237	15	16	#236	GPIOX.BIT8	236		4
			3V3	17	18	#233	GPIOX.BIT5	233		5
12	235	GPIOX.BIT7	#235	19	20	GND				
13	232	GPIOX.BIT4	#232	21	22	#231	GPIOX.BIT3	231		6
14	230	GPIOX.BIT2	#230	23	24	#229	GPIOX.BIT1	229		10
			GND	25	26	#225	GPIOY.BIT14	225		11
	207	I2CB_SDA	SDA2	27	28	SCL2	I2CB_SCL	77		
21	228	GPIOX.BIT0	#228	29	30	GND				
22	219	GPIOY.BIT8	#219	31	32	#224	GPIOY.BIT13	224		26
23	234	GPIOX.BIT6	#234	33	34	GND				
24	214	GPIOY.BIT3	#214	35	36	#218	GPIOY.BIT7	218		27
		ADC.AIN1	AIN1	37	38	1V8	1V8			
			GND	39	40	AIN0	ADC.AIN0			

Fig. 2.5