ENSC351

GROVE PIR MOTION SENSOR

Kenton Dooley
Kenvir Sidhu
Jaspreet Shergill

This document guides the user through

1. Connecting Grove PIR Motion Sensor to BBG
2. Sensor and command line interface
3. C code for sensor

Table of Contents

Formatting

1. Commands for the host Linux's console are show as:
   ```
   (host)$ echo "Hello PC world!"
   ```
2. Commands for the target (BeagleBone) Linux's console are shown as:
   ```
   (bbg)$ echo "Hello embedded world!"
   ```
3. Almost all commands are case sensitive.

1. **Connect Grove PIR Motion Sensor to Camera**

    1. Connect motion sensor to BBG

       Mapping (sensor to BBG):

       GND (sensor) - GND (BBG)
       VCC (sensor) - 3V3 (BBG)
       NC (sensor) - SDA (BBG)
       D1 (sensor) - SCL (BBG)

2. **Motion Sensor via Command Line**

    2.1 Configure GPIO

    1. On the target, run the following command:
       ```
       (bbg)$ config-pin p9_19 gpio
       ```

       Upon entering the command current mode should be set to GPIO

       ```
       Current mode for P9_19 is:     gpio
       ```

    2.2 Reading Output

    2. To ensure proper functionality of motion sensor, read the output value of the sensor
       ```
       (bbg)$ cd /sys/class/gpio/gpio13
       (bbg)$ cat value
       ```

       A one will be concatenated to the screen when motion is detected and a zero will be concatenated to the screen when motion is not detected.

    3. **Troubleshooting:**
       When reading in the value, if the value output remains a zero (while demonstrating motion and not demonstrating motion) ensure current mode of pin P9_19 is set to gpio and all connections to the BBG are intact.

       To check current mode enter the following command:
       ```
       (bbg)$ config-pin -q P9_19
       ```

3. C code to test PIR Motion Sensor Functionality

1. Here is an example of a simple C program to sample the motion sensor every 2s and have the program terminate using the user button

motion.h

```
#ifndef _PIRMOTION_H
#define _PIRMOTION_H

#define FILE_TO_READ "/sys/class/gpio/gpio13/value"

void initSetup();
void sleepForMs(long long delayInMs);
void Start_Detecting_Motion(void);
void Stop_Detecting_Motion(void);

int readFromFileToScreen(char *fileName);

#endif
```

motion.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <pthread.h>
#include <unistd.h>
#include "PIRmotion.h"

static pthread_t idRead;
bool isDoneRunning = false;

void sleepForMs(long long delayInMs){
      const long long NS_PER_MS = 1000 * 1000;
      const long long NS_PER_SECOND = 1000000000;
      long long delayNs = delayInMs * NS_PER_MS;
      int seconds = delayNs / NS_PER_SECOND;
      int nanoSeconds = delayNs % NS_PER_SECOND;
      struct timespec reqDelay = {seconds, nanoSeconds};
      nanosleep(&reqDelay, (struct timespec *) NULL);
}
```

```
static void runCommand(char* command){

	// Execute the shell command (output into pipe)
	FILE *pipe = popen(command, "r");

	// Ignore output of the command; but consume it
	// so we don't get an error when closing the pipe.
	char buffer[1024];
	while (!feof(pipe) && !ferror(pipe)) {
	if (fgets(buffer, sizeof(buffer), pipe) == NULL){
		break;
	}
	// printf("--> %s", buffer);
	}

	// Get the exit code from the pipe; non-zero is an error:
	int exitCode = WEXITSTATUS(pclose(pipe));
	if (exitCode != 0){
	perror("Unable to execute command:");
	printf(" command:   %s\n", command);
	printf(" exit code: %d\n", exitCode);
	}

}

void initSetup()
{
	runCommand("config-pin p9_19 gpio");
	runCommand("config-pin p8.43 gpio");
}

int readFromFileToScreen(char *fileName){
	FILE *pFile = fopen(fileName, "r");
	if (pFile == NULL){
	printf("ERROR: Unable to open file (%s) for read\n", fileName);
	exit(-1);
	}

	// Read string (line)
	const int MAX_LENGTH = 1024;
	char buff[MAX_LENGTH];
	fgets(buff, MAX_LENGTH, pFile);

	// Close
	fclose(pFile);
	return atoi(buff);
	// printf("Read: '%s\n", buff);
}
```

```
void *CaptureThread (void *_)
{
	while(!isDoneRunning){
	int motionValue = readFromFileToScreen(FILE_TO_READ);
	if (motionValue == 1){
		printf("Motion Detected!\n");
		sleepForMs(2000);
	}
	else{
		printf("Motion not Detected!\n");
		sleepForMs(2000);
	}
	}
	return NULL;
}

void Start_Detecting_Motion()
{
	pthread_create(&idRead, NULL, &CaptureThread, NULL);
}

void Stop_Detecting_Motion()
{
	isDoneRunning = true;
	printf("Thread exiting!\n");
	pthread_join(idRead, NULL);

}
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <pthread.h>
#include <unistd.h>
#include "PIRmotion.h"

#define USER_BUTTON_VALUE_FILE "/sys/class/gpio/gpio72/value"

int main()
{
	initSetup();
	Start_Detecting_Motion();
	printf("Press the user button to quit.\n");
	int userButtonState = readFromFileToScreen(USER_BUTTON_VALUE_FILE);
	while (true){

	sleepForMs(100);
	userButtonState = readFromFileToScreen(USER_BUTTON_VALUE_FILE);
	if (userButtonState == 0){
		break;
	}
	}
	Stop_Detecting_Motion();
}
```

References

https://opencoursehub.cs.sfu.ca/bfraser/grav-cms/ensc351/links/files/2018-student-howtos/GrovePIRMotionSensorAndLiveStreaming.pdf