Jan Niemiec
Patryk Woźniak

Microprocessor Technology Laboratory
Project documentation

Wii-like game controller using MPU6050

1. Introduction

The purpose of the project was to configure MPU6050 sensor to use it as a motion game controller, using Kinetis FRDM board for communication with PC. MPU6050 is a motion tracking device that combines 3-axis gyroscope and 3-axis accelerometer. The device uses I2C interface to communicate with the microcontroller.

2. Kinetis application

Microcontroller application was designed using CodeWarrior IDE and Processor Expert expansion. Processor Expert lets the user easily generate configuration code for processor components such as Timers, GPIOs or I2C and UART modules. Most significant components used in our project are:

- TimerUnit_LDD - to generate interrupt to change state in the state machine
- I2C_LDD - to communicate with MPU6050
- ConsoleIO - to send data to PC using UART
- BitIO_LDD - to detect whether or not is shoot button pressed

Timer Configuration:

Name	Value	Details
Module name	TPM0	TPM0
Counter	TPM0_CNT	TPM0_CNT
Counter direction	Up	
Input clock source	Internal	
Counter frequency	655.36 kHz	655.360 kHz
Counter restart	On-overrun	
Overrun period	Auto select	100 ms
Interrupt	Enabled	
Channel list	0	6 available channels
Initialization		
Auto initialization	no	

TPM0 counter is being used. It runs at 6.55.360 kHz frequency and generates interrupt every 50ms.

I2C Configuration:

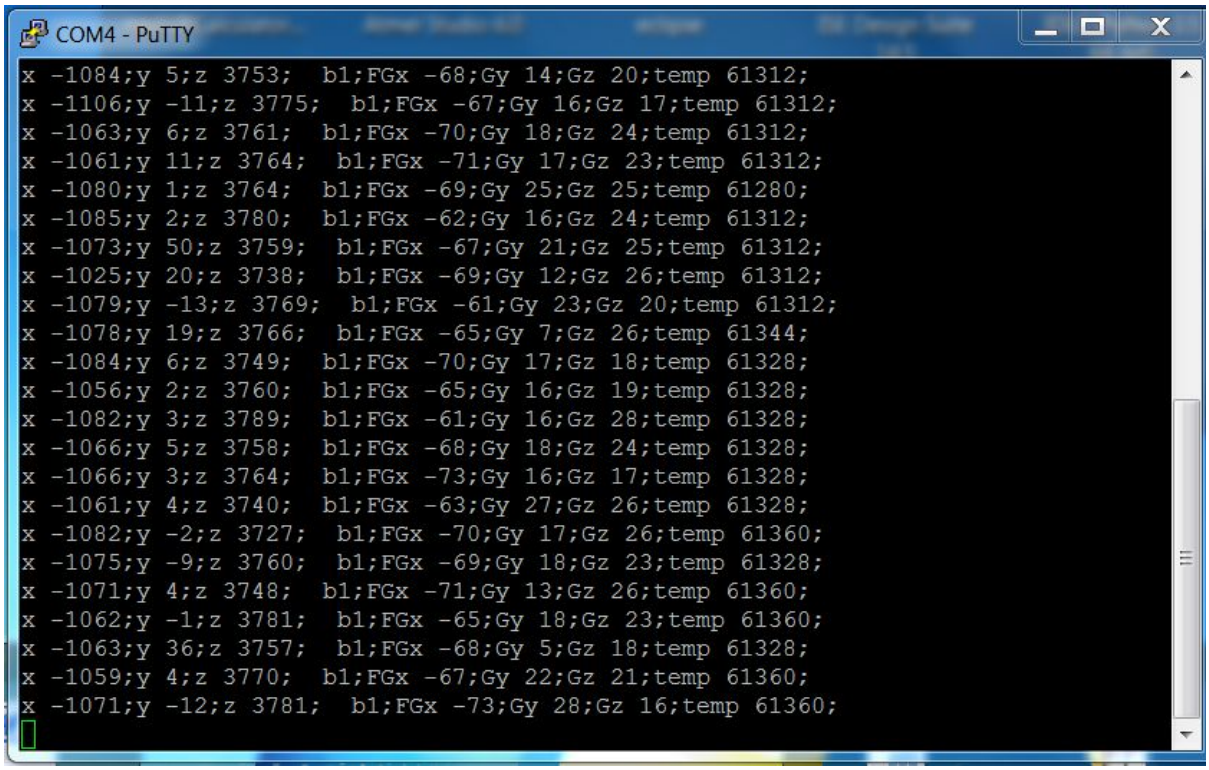
Name	Value	Details
I2C channel	I2C0	I2C0
Interrupt service	Enabled	
▲ **Settings**		
Mode selection	MASTER	
▲ **MASTER mode**	Enabled	
▲ **Initialization**		
Target slave address init	68 H	
› **SLAVE mode**	Disabled	
▲ **Pins**		
▲ **SDA pin**		
SDA pin	LCD_P29/CMP0_IN3/PTC9/I...	LCD_P29/CMP0_IN3/PTC9/I2C...
▲ **SCL pin**		
SCL pin	LCD_P28/CMP0_IN2/PTC8/I...	LCD_P28/CMP0_IN2/PTC8/I2C...

We use I2C0 module of Kinetis kl46z board. We set the processor as Master device and provide MPU6050 address as slave device address. PTC9 pin is used as SDA pin, while PTC8 is SCL pin.

ConsoleIO (UART module) Configuration:

Name	Value	Details
Data width	8 bits	
Parity	None	
Stop bits	1	
Loop mode	Normal	
Baud rate	115200 baud	113975.652 baud
Wakeup condition	Idle line wakeup	
Stop in wait mode	no	
Idle line mode	Starts after start bit	
Transmitter output	Not inverted	
Receiver input	Not inverted	
Break generation length	10/11 bits	
▲ **Receiver**	Enabled	
RxD	TSI0_CH2/PTA1/UART0_RX/...	TSI0_CH2/PTA1/UART0_RX/T
RxD pin signal		
▲ **Transmitter**	Enabled	
TxD	TSI0_CH3/PTA2/UART0_TX/...	TSI0_CH3/PTA2/UART0_TX/T
TxD pin signal		

UART communication parameters are set as follow: 115200 Baud rate, no parity bit, 1 stop bit, 8 bits data width. Pin PTA1 and PTA2 are used as Rx and Tx accordingly. Those pins are connected to USB port of the programmator featured on the board and therefore can be used to send and receive data from PC.



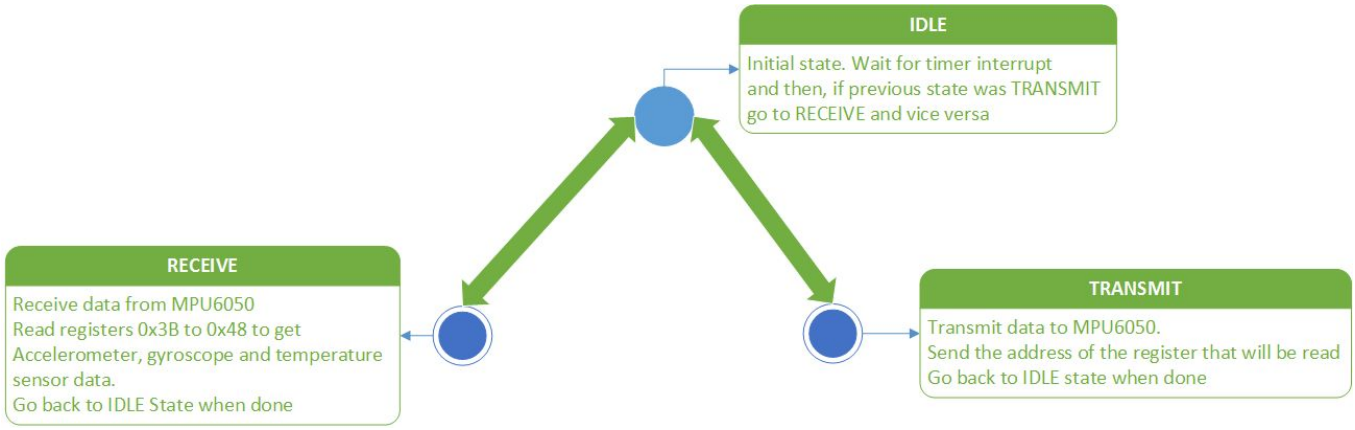
MPU initialization process:

```
void MPUinit(void){
	OutData[0]=MPU6050_RA_SMPLRT_DIV; //Sample Rate Divider register addres
	OutData[1]=0x07;                  //data to be send to register

	//Sets sample rate to 8000/1+7 = 1000Hz
	Error = CI2C1_MasterSendBlock(MyI2CPtr, &OutData, 2, LDD_I2C_NO_SEND_STOP);
	while (!DataTransmittedFlg);
	DataTransmittedFlg = FALSE;

	OutData[0]=MPU6050_RA_GYRO_CONFIG; //Gyroscope configuration register address
	OutData[1]=0b00001000;             //data to be send to register

	//Disable gyro self tests, scale of 500 degrees/s
	Error = CI2C1_MasterSendBlock(MyI2CPtr, &OutData, 2, LDD_I2C_NO_SEND_STOP);
	while (!DataTransmittedFlg);
	DataTransmittedFlg = FALSE;

	OutData[0]=MPU6050_RA_PWR_MGMT_1; //Power Management 1 register address
	OutData[1]=0b00000010;            //data to be send to register

	//Sets clock source to gyro reference w/ PL
	Error = CI2C1_MasterSendBlock(MyI2CPtr, &OutData, 2, LDD_I2C_NO_SEND_STOP);
	while (!DataTransmittedFlg);
	DataTransmittedFlg = FALSE;


}
```

Comments included in the code explain the process.

Data acquisition:

Sensors' data is collected using state machine and polling. State machine diagram with state descriptions is shown below.



Transition from **IDLE state** to RECEIVE or TRANSMIT is done in timer interrupt handler function:

```
void Timer_Interrupt_CB(void)
{
  if(measuring == IDLE)
  {
    if(measuring_last == TRANSMIT)
      measuring = RECEIVE;
    else if(measuring_last == RECEIVE)
      measuring = TRANSMIT;
  }
}
```

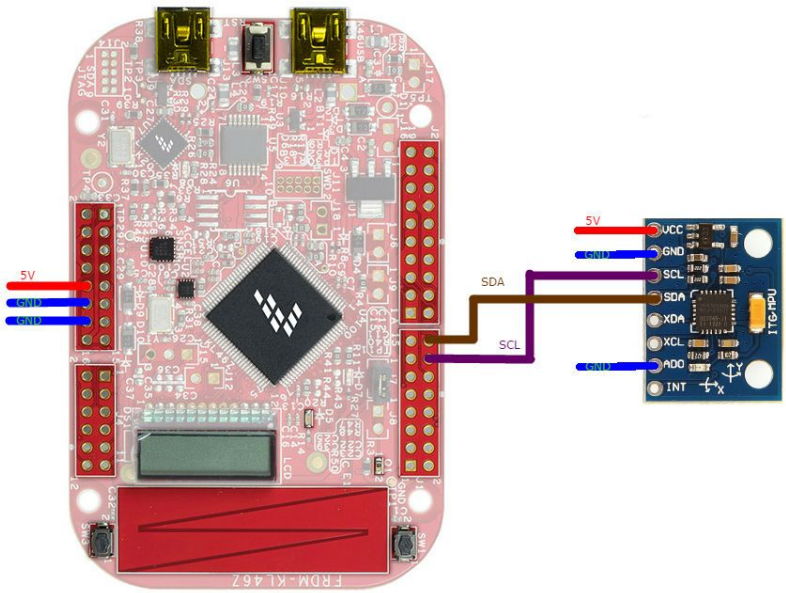
In **TRANSMIT state** ACCEL_XOUT_H (0x3B) register address is being send to the MPU to let it know that we will be reading registers starting from there.

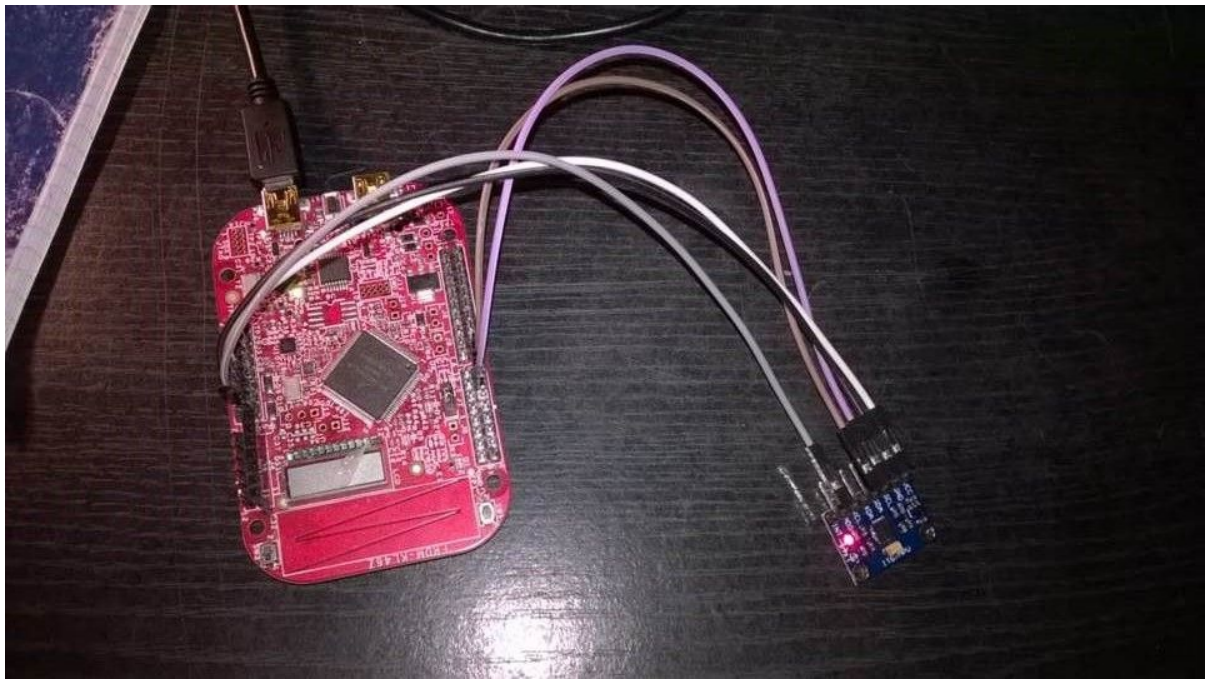
```
if(measuring == TRANSMIT)
{
  measuring = IDLE;
  measuring_last = TRANSMIT;

  OutData[0] = MPU6050_RA_ACCEL_XOUT_H;
  Error = CI2C1_MasterSendBlock(MyI2CPtr, &OutData, 1, LDD_I2C_NO_SEND_STOP);
  while (!DataTransmittedFlg);
  DataTransmittedFlg = FALSE;
```

In **RECEIVE state** data is being collected. We collect data from both accelerometer and gyroscope, and also from temperature sensor also included in MPU6050. Acquired data contains of 7 16bit values, but since I2C bus provides 8bit values, we received 14 bytes and we need to join appropriate bytes together. Example below.

```
else if(measuring == RECEIVE)
{
  measuring = IDLE;
  measuring_last = RECEIVE;

  //Read sensors data
  Error = CI2C1_MasterReceiveBlock(MyI2CPtr, &InData, MPU_BUFFER_SIZE, LDD_I2C_SEND_STOP);
  while (!DataRecivedFlg);
  DataRecivedFlg = FALSE;

  mma845_tmp = InData[1] | (InData[0] << 8); //join bytes toggether to get the measure dvalue
```

Schematic:

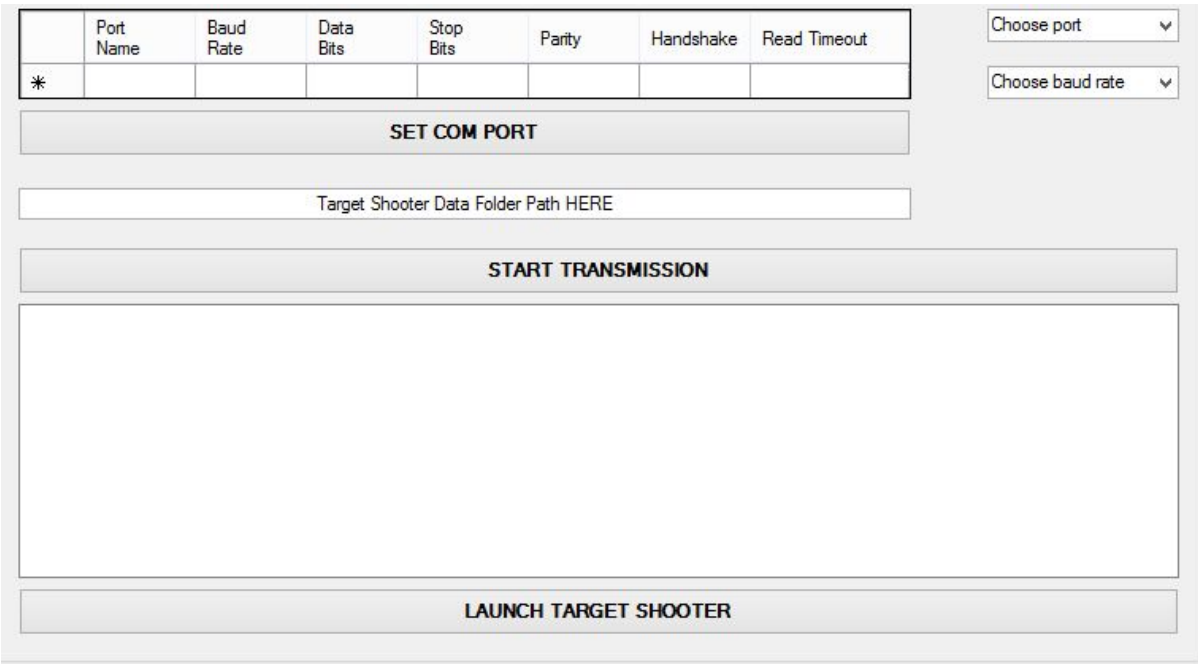

5V
GND
GND
SDA
SCL
5V
GND
VCC
GND
SCL
SDA
XDA
XCL
ADO
INT
ITG-MPU
FRDM-KL46Z



3. Client application.

UART Data Provider

UART Data Provider is responsible for downloading and processing data from Kinetis KL46Z to PC via UART communication.



UART Data Provider User Interface

User is able to choose COM port and set baud rate. Before the start of transmission user has to confirm his settings by clicking the "**SET COM PORT**" button and put destination path into textbox named "**Target Shooter Data Folder Path HERE**". If transmission ended successfully, data lines should appear in listbox below. Then the last thing to do is pressing the **LAUNCH TARGET SHOOTER** to enjoy Target Shooter.

Data processing.

UART Data Provider splits received data to smaller parts (integer values) and saves them to **CSV** file into **Target Shooter Resources** folder. Then **Unity Application** reads saved file as **TextAsset** and generates **List** of values. Finally **WeaponController** component uses these values to move/reload action.

Target Shooter Application

Target Shooter was created in Unity Engine. Main purpose of game is to shoot as many targets as user can within 1 minute. To steer the weapon user needs Kinetis KL46Z with application mentioned in point 2. Kinetis KL46Z has to be plugged into USB port in user computer and UART Data Provider transmission must be set before launch.

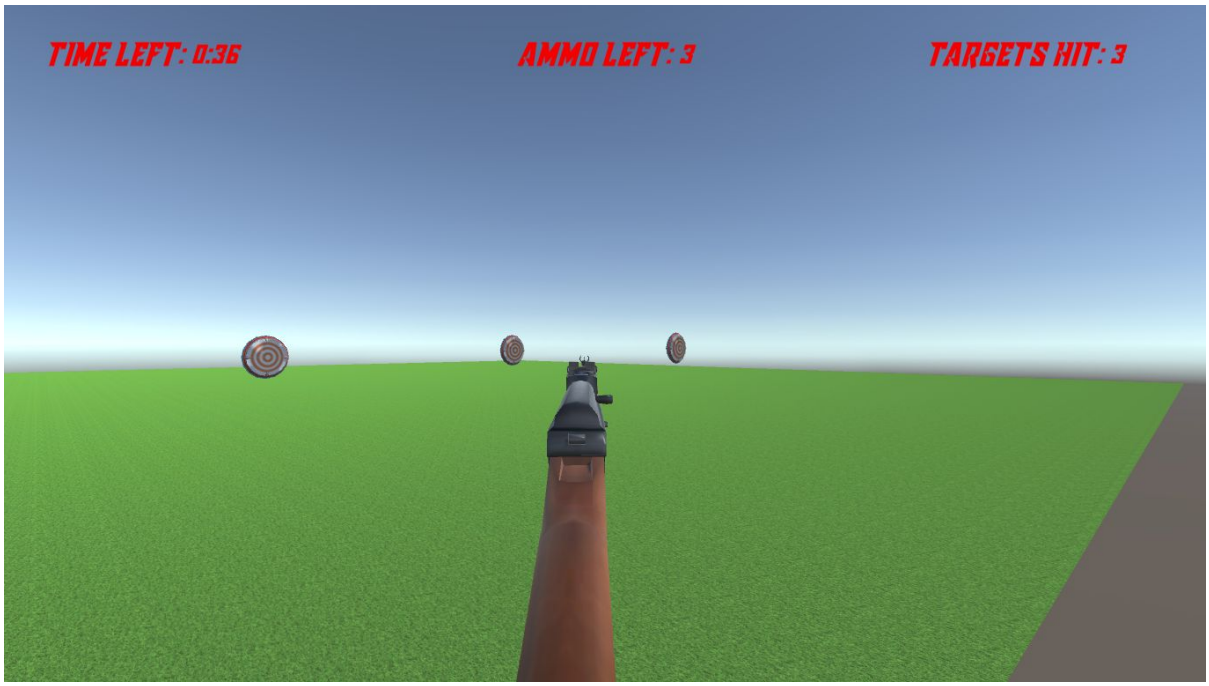
Start game screen

To start game user should press ‘Space’ button.

To change weapon position user should rotate KL46Z left or right, raise to jump.

Reload action

To reload weapon user should tilt KL46Z forward, works only if clip is empty (**Ammo Left** counter shows 0).



Gameplay Screen