MSE 450
Real-Time and Embedded Control Systems

Project Report

3-Axis Accelerometer Game Controller

Summer 2023

Group 3

August 1, 2023

Pardeep Bhattal 301397266
Parminder Gill 301394377
Yuvraj Ghuman - 301397105
Imroz Saran - 301378518

Abstract

This final project report presents the objectives, methodologies, and outcomes of a comprehensive project aimed at developing an accelerometer-based game controller. The project involved hardware interfacing, signal processing, and software development to read 3-axis acceleration, calculate roll and pitch angles, control on-board LEDs, generate sinusoidal waves, and stream data to control a space invaders python game. The onboard LIS3DSH triple axis accelerometer from the STM32 board is used to control the game. Overall, this project provided the team with hands-on experience in integrating hardware and software components and analyzing the effects of altering input and control parameters on the performance of the accelerometer-based game controller.

Table of Contents:

1. Introduction

In this final project, our central focus lies in the development of an accelerometer-based game controller using the LIS3DSH chip. This project aims to integrate hardware interfacing, signal processing, and software implementation to achieve a fully functional and immersive gaming experience.

The primary objectives of this project are twofold: firstly, to read 3-axis acceleration data from the LIS3DSH accelerometer chip and calculate roll and pitch angles accurately, and secondly, to utilize this angle data to control on-board LEDs and generate sinusoidal waves for visual feedback and enhanced gaming interactions. Additionally, we aim to stream the angle data through USB to control a Python game, enabling players to maneuver the game character using physical movements. To accomplish these objectives, we will employ the STM32Cube development environment for programming the microcontroller and interacting with the LIS3DSH chip and other peripherals.

The project can be divided into several main components, including reading and processing accelerometer data, controlling on-board LEDs to reflect the angles, generating sinusoidal waves for visual feedback, and establishing a communication link with the Python game for player control.

Throughout the report, we will provide detailed explanations of each component's implementation. As we progress through the project, we expect to gain valuable insights into embedded control systems, signal processing, and game development, while further enhancing our understanding of real-time applications and practical implementations.

2. Implementation Details

In this section, we provide an in-depth overview of the key implementation details of our accelerometer-based game controller project. We delve into the technical aspects of how we accomplished each objective and integrated various hardware and software components to achieve a fully functional system.

2.1: Hardware Setup

The hardware setup for the accelerometer-based game controller involves the following components:

STM32 Microcontroller: The STM32 microcontroller serves as the brain of the system, responsible for processing data from the accelerometer and generating control signals for LEDs and the DAC.

LIS3DSH Accelerometer: The LIS3DSH accelerometer is a three-axis digital accelerometer capable of measuring linear acceleration on the X, Y, and Z axes.

LEDs: The game controller features two on-board LEDs, which provide visual feedback to the user by changing their brightness based on the calculated pitch and roll angles.

DAC (Digital-to-Analog Converter): The DAC is utilized to convert digital data into analog voltage signals. It generates sinusoidal waves with amplitudes and frequencies proportional to the roll and pitch angles, respectively.

USB Interface: The game controller incorporates a USB interface to stream the calculated pitch and roll angles to the computer host. The USB communication uses the CDC (Communication Device Class) interface, which allows the controller to emulate a virtual COM port.

Timer 6: This timer is configured in interrupt mode to generate periodic interrupts every 20ms. The interrupt callback function HAL_TIM_PeriodElapsedCallback() is defined to handle these timer interrupts. Inside this function, the accelerometer's XYZ coordinates are read, and the pitch and roll angles are calculated using the calcAngles() function. The pitch and roll angles are then converted to string format and transmitted over USB to the computer host using the CDC (Communication Device Class) interface.

Timer 4: Timer 4 is set up to generate PWM (Pulse-Width Modulation) signals for driving all four on-board LEDs. The LEDs' brightness is modulated based on the pitch and roll angles to provide visual feedback to the user. The PWM duty cycle for each LED is calculated based on the pitch and roll angles and then applied using the __HAL_TIM_SET_COMPARE() function.

Timer 2: This timer is used for generating a sinusoidal waveform for visual feedback, simulating the rolling and pitching motion of the game controller. The timer's ARR (Auto-Reload Register) value is adjusted based on the pitch angle to change the frequency of the sine wave. The DAC is triggered by Timer 2's TRGO (Trigger Output) event to generate the analog output waveform.

2.2: Reading 3-Axis Acceleration and Calculating Angles

SPI communication protocol is used to read the data from the onboard LIS3DSH accelerometer. The accelerometer data reading and angle calculation occur in the HAL_TIM_PeriodElapsedCallback() function, triggered by Timer 6's periodic interrupts. The function reads the accelerometer's XYZ coordinates using the ACCELERO_GetXYZ() function and stores the data in the pData array. The calcAngles() function then calculates the pitch and roll angles based on the accelerometer data stored in pData. The pitch angle (pAngles[0]) represents the tilting motion along the X-axis, and the roll angle (pAngles[1]) represents the tilting motion along the Y-axis. The calculation to calculate the pitch and roll uses the atan2 function from the math.h standard C library.

2.3: Controlling On-Board LEDs via Angle

The project utilizes Timer 4 (htim4) to control the brightness of the four on-board LEDs based on the calculated pitch and roll angles. The duty cycles for each PWM channel (TIM_CHANNEL_1 to TIM_CHANNEL_4) are calculated based on the pitch and roll angles, and they are adjusted using the __HAL_TIM_SET_COMPARE() function. The pitchLeftPWM, pitchRightPWM, rollFwdPWM, and rollBackPWM variables store the calculated PWM duty cycles for the corresponding LED channels, resulting in changes in LED brightness as the controller is tilted.

2.4: Generating Sinusoidal Waves Proportional to Angles

The DAC (Digital-to-Analog Converter) is employed to generate sinusoidal waves with amplitudes and frequencies proportional to the roll and pitch angles, respectively. The setSineData() function calculates the amplitude and frequency of the sine wave based on the current pitch and roll angles. The amplitude is scaled to the DAC's 12-bit resolution, and its value is adjusted proportionally to the roll angle to provide visual feedback to the user. The frequency is adjusted proportionally to the pitch angle, modifying the ARR (Auto-Reload Register) value of Timer 2, allowing the DAC to generate the output at the desired frequency, simulating the pitching motion of the game controller.

2.5: Streaming Angles through USB and Game Control

The game controller streams the calculated pitch and roll angles to the computer host using the USB interface with the CDC (Communication Device Class) protocol. The USB communication is set up using the MX_USB_DEVICE_Init() function. Inside the HAL_TIM_PeriodElapsedCallback() function (triggered every 20ms by Timer 6), the pitch and roll angles are converted to string format and transmitted over USB to the computer host using the CDC interface. The angles are sent as a string in the format "pitch,roll\r\n". The computer host can receive these angle values through the virtual COM port created by the USB interface. These angles can be used to control the Python game, providing a dynamic and interactive gaming experience based on the player's tilting motions.

2.6: Flight Simulator

As a bonus exercise the team used the 3-axis accelerometer to control a flight simulator. Additionally, a speaker was implemented to deliver feedback relative to the tilt of the microcontroller. The aim was to use the real-time sensor data from the board to control the flight simulator's aircraft, providing an immersive and interactive flying experience. To begin, we opened the FlightGear simulator and made the necessary configuration adjustments to enable communication with the STM32F407G-DISC board.

Real-time sensor data from the board is read by a python script and the aircraft movements within the simulator. Tilting the board adjusted the aircraft's roll and pitch, offering an interactive and engaging flying sensation.

2.7: Testing and Debugging

Throughout the development process, we tested each component independently to ensure its functionality. We then integrated the components and performed thorough testing to verify the complete system's functionality. Debugging was carried out to address any issues or discrepancies that arose during testing.

3. Conclusion

The STM32-based accelerometer-based game controller project successfully achieved its objectives, offering a great gaming experience through hardware integration and software control algorithms.The hardware setup, including the STM32 microcontroller, LIS3DSH accelerometer, LEDs, and DAC, facilitated interactive control in response to user tilting motions. Accurate 3-axis acceleration reading and angle calculation provided crucial orientation data for controlling on-board LEDs, visually indicating the controller's position. Generating sinusoidal waves based on roll and pitch angles added dynamic audio feedback, immersing users further in the gaming environment. Streaming angles through USB enabled seamless communication with the Python game, empowering players to control the game character using physical movements.

This project provided valuable hands-on experience in embedded systems development, sensor interfacing, real-time control algorithms, and USB communication. It showcased the potential of accelerometer-based controllers in interactive human-computer interactions and gaming. Overall, the successful completion of this project strengthens our skills in creating user-friendly applications and fuels our passion for exploring future innovative ventures in the field of real-time and embedded control systems.

4. Appendix

Project GitHub Repository: https://github.com/PardeepSB/DAMCS