Maritime Safety and Stability Alert System (MSSAS)

EE128 Final Project

PART 1. Project Description

PART 2. System Design

PART 3. Implementation Details

PART 4. Testing & Evaluation

PART 5. Discussion

PART 6. Roles and Assignment Distributions

PART 7. Conclusion

Contributors:

Cristian Molano
cmola002@ucr.edu

Johan Gonzalez
jgonz563@ucr.edu

Martin Mejia
mmeji049@ucr.edu

Part 1. Introduction

Maritime Safety and Stability Alert System (MSSAS) is an innovative project aimed at developing a comprehensive safety solution for boats, providing real-time visual and audio alerts in response to excessive tilting. This system leverages modern microcontroller technology, specifically the K64F and Arduino Uno, to detect and warn about potential capsizing events due to adverse weather conditions or control malfunctions. The project focuses on designing and prototyping a robust safety system that can be integrated into various maritime vessels, enhancing their operational safety. With a few modifications, it can also serve other applications outside of the maritime industry including aerial applications.

Complexities

- Serial Communication: Utilizes UART for data transmission between the K64F microcontroller and the Arduino Uno.
- GPIO Pins: Essential for interfacing with various components like LEDs, buzzers, and LCD displays.
- Magnetometer Data Processing: Involves sampling the Y-axis data from the K64F magnetometer to detect boat tilt.
- Real-time Data Processing: The Arduino Uno processes incoming data to determine the boat's stability status and triggers appropriate warnings.

Extra Parts

- GPS Module: Provides real-time location tracking to enhance situational awareness.
- Water Level Sensor: Detects potential water ingress, adding another layer of safety.
- LCD Display: Shows visual warnings such as “Tilting to Right” or “Boat is Drowning”.
- 90dB Buzzer: Emits audio warnings to prompt immediate action.
- Red and Green LEDs: Indicate the stability status of the boat visually.
- 5V 40A Power Supply: Ensures the system is adequately powered.

By incorporating these complexities and additional components, the Maritime Safety and Stability Alert System (MSSAS) aims to provide a robust and reliable safety solution for maritime vessels, significantly enhancing their ability to respond to potential capsizing events.

Part 2. System Design

Figure 1 is a picture taken during the development of this project. Important to note is that most components at this point have been implemented and the core functionality of the project is working. The two microcontrollers shown are utilizing UART communication to transmit data to each other regarding the tilt angles of the system itself. The detection system is in the enclosure which simulates a boat in open sea.

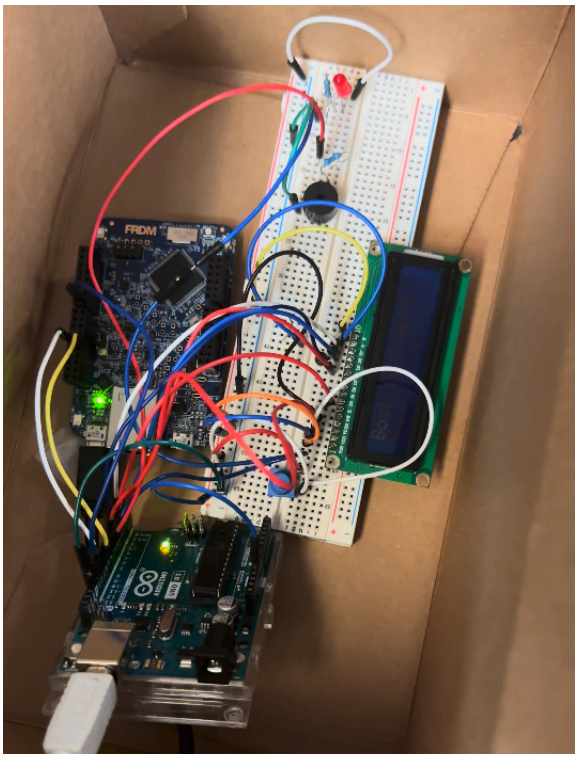

Figure 1. Photo of Setup During Development

Implementation Methods

Tilt Detection and Warning System

- Utilizes the K64F microcontroller's magnetometer to sample Y-axis data, detecting the boat's tilt.
- Processes this data and transmits it via UART to the Arduino Uno, which then triggers appropriate visual and audio warnings.

Visual and Audio Alerts

- Visual Alerts: An LCD display shows messages such as “Tilting to Right” or “Boat is Drowning” based on the tilt severity.

- Audio Alerts: A 90dB buzzer emits sounds to prompt immediate action in case of excessive tilting.

Real-time Monitoring and Data Processing

- Continuously monitors the boat's tilt angle.
- Provides real-time updates through the serial monitor, allowing crew members to track the stability status of the boat.

GPS Integration

- A GPS module is integrated to provide real-time location data.
- The system logs the boat's position at regular intervals, aiding in tracking and rescue operations if necessary. Whenever the boat is capsized, the GPS data coordinates will be transmitted.

Water Level Sensor

- A water level sensor is installed to detect any water ingress into the boat.
- The sensor data is transmitted to the Arduino, which triggers alerts if the water level exceeds a predefined threshold.

Block Diagrams and Flowcharts

- **K64F Microcontroller:**
    - Magnetometer sensor connected to sample Y-axis data.
    - UART communication setup to send data to Arduino Uno.
- **Arduino Uno:**
    - Receives data from K64F.
    - Controls the LCD display, LEDs, and buzzer based on received data.
    - Processes GPS and water level sensor data to provide comprehensive safety alerts.
- **Power Supply:**
    - 5V 40A power supply ensures adequate power to the entire system. This power supply may actually be overkill for a project like this, but it was provided by UCR lab department, so this will do.

Figure 2 shows the system block diagram in which all relevant components of this project are connected and how they work together to achieve the complete functionality.

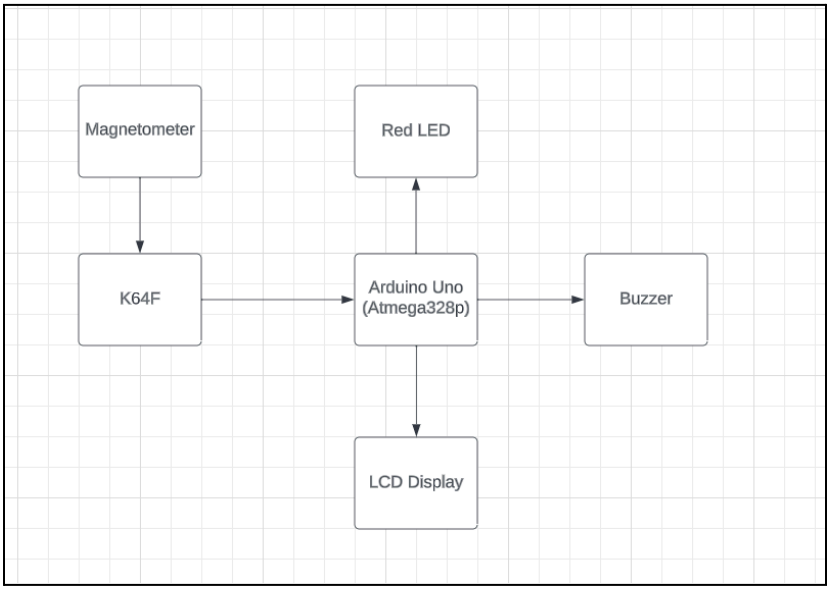

Figure 2. Hardware Block Diagram

Flowchart of the K64F Microcontroller

1. Initialization:
    - Set up UART communication.
    - Initialize magnetometer sensor.
2. Data Sampling:
    - Continuously read Y-axis data from the magnetometer and GPS module.
    - Convert data into a string format.
3. Data Transmission:
    - Send the string data via UART to the Arduino Uno.

Figure 3 shows the system flowchart of the K64F microcontroller in the system. This component is constantly checking the magnetometer data and providing updates to the Arduino Uno. This is achieved by checking the values of all individual axes in the magnetometer data and writing this data via a string into a buffer. This is done by programming all necessary code logic utilizing the registers and hex codes. Finally, the magnetometer data is passed along via a special function that is called and transmitted to the Arduino Uno. With this, the data can then be processed by the Arduino and used to provide feedback on vehicle system stability.

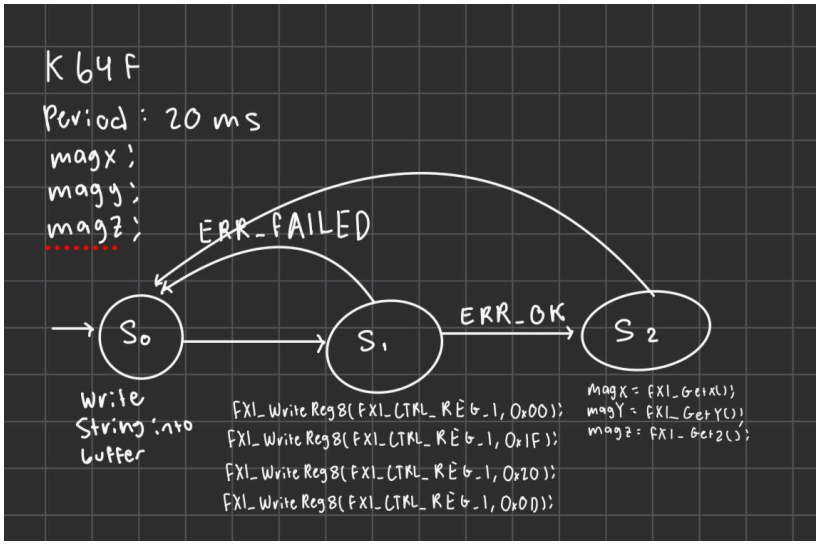

Figure 3. K64F System Flowchart

Flowchart of the Arduino Uno

1. Initialization:
    - Set up serial communication.
    - Initialize LCD display, LEDs, and buzzer.
2. Data Reception:
    - Continuously read data from the K64F via UART.
3. Data Processing:
    - Convert received data into an integer representing the tilt angle.
    - Read GPS and water level sensor data, store and transmit to Arduino
    - Determine the boat's stability status based on the tilt angle.
4. Trigger Alerts:
    - Display appropriate messages on the LCD.
    - Activate LEDs and buzzers based on the severity of the tilt.

Figure 4 shows the flowchart of the detection system using the magnetometer data. Once the data is transmitted and received by the Arduino microcontroller, it can be processed and used to send warnings accordingly. The X and Y axes are crucial in correctly calculating and detecting a capsizing event. Initially, the data is being transmitted via a buffer. This data is then processed using specified threshold values that are set according to the real world scenario. In this case the team is showcasing a boat in open waters. If within the “safe” threshold, the display will simply say the boat is “stable”. If the magnetometer data shows that the boat is leaning past the safe threshold, it will give a warning in the display and buzzer prompting immediate action. If the boat completely capsizes, this will trigger the corresponding messages as well. All together, this system is useful in detecting when a boat is about to tip over and when it is completely capsized.

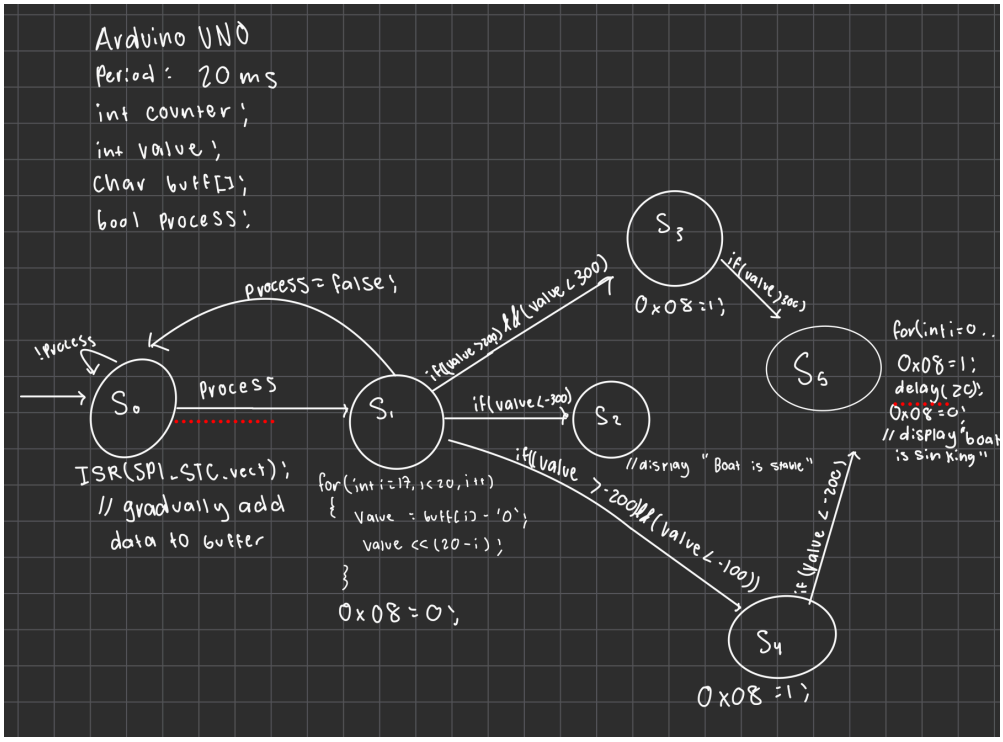

Figure 4. Detection System Flowchart

Explanation

The system begins operation once powered on. The K64F microcontroller continuously samples Y-axis data from the magnetometer to detect the boat's tilt. This data is transmitted via UART to the Arduino Uno, which processes the information to determine the boat's stability status.

- **Automatic Mode:**
  - If the tilt angle exceeds the safe threshold, the system enters an alert state.
  - The LCD display shows warning messages, and the buzzer emits sound alerts.
  - Red LEDs flash to indicate instability or danger. This is when GPS coordinates are gathered and transmitted to the Arduino in order to be transmitted to an emergency service.
- **Manual Override:**
  - The system can be manually overridden to stop automatic alerts.
  - Allows for maintenance or manual inspection without triggering false alarms.
- **Real-time Monitoring:**
  - The serial monitor displays the current tilt angle, providing real-time statistics for the crew to assess the situation accurately.
  - GPS data provides real-time location tracking for enhanced safety and backup protocols in case full capsizing occurs.
  - Water level sensor alerts if water ingress is detected.

This comprehensive system design ensures that the Maritime Safety and Stability Alert System (MSSAS) provides timely and accurate warnings, enhancing the safety and stability of maritime vessels. Because this system is portable and

Part 3. Implementation & Circuit Schematics

Schematics

The schematic shown in figure 5 details all wiring and component implementations for this project. It can be seen how the K64F operates in conjunction with the Arduino Uno to provide it with the necessary magnetometer and accelerometer data. The display and buzzers are wired using traditional techniques using pins 9, 5, 4, 3, and 2. The necessary resistors and capacitors are also shown. The GPS module which was implemented in the finalized project is connected directly to the Arduino as a separate component. Its values get read continuously and gathered in a data array which then gets sent over.

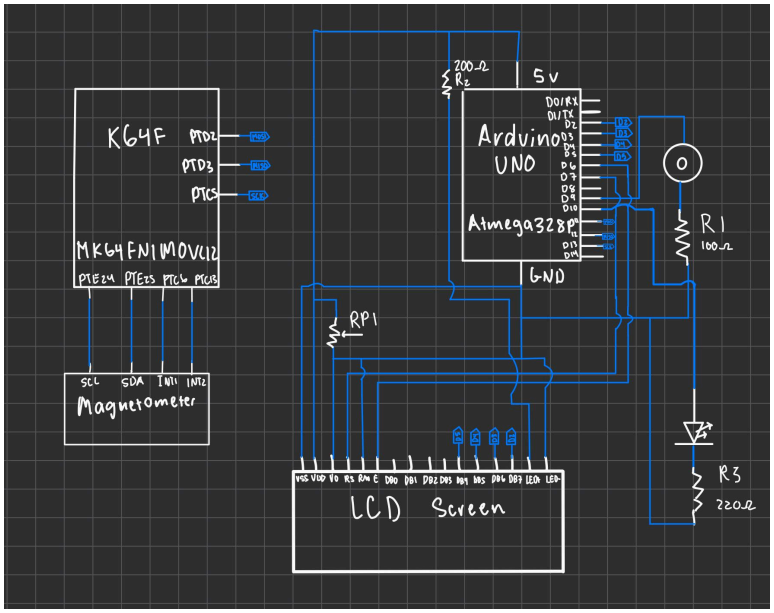

Figure 5. Circuit Schematic

Code

There were two development environments used to build this project. For programming the K64F microcontroller, Kinetis Design Studio IDE was used. This is where we programmed the K64F to set up UART communication protocols. This is also where we gathered the magnetometer and accelerometer data into a string which was then sent to the Arduino Uno.

For programming the Arduino Uno, Arduino IDE was used. This is where the gathered data was used to control the display and buzzer according to the values and what they represented. We chose this implementation method because it was the most straightforward and organized way to maintain the two microcontroller programs. Below, the code for both portions of the project can be found.

Kinetis Design Studio IDE Program (K64F Code):

```
#include "Cpu.h"
#include "Events.h"
#include "Pins1.h"
#include "FX1.h"
#include "GI2C1.h"
#include "WAIT1.h"
#include "CI2C1.h"
#include "CsIO1.h"
#include "IO1.h"
#include "MCUC1.h"
#include "SM1.h"
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include "PDD_Includes.h"
#include "Init_Config.h"

unsigned char write[512];
int main(void)

{     PE_low_level_init();
      uint32_t delay;
      uint8_t ret, who;
      int16_t magX, magY, magZ;

      int len;
      LDD_TDeviceData *SM1_DeviceData;
      SM1_DeviceData = SM1_Init(NULL);

      printf("Hello\n");

      FX1_Init();

      for(;;) {
            // get WHO AM I values
            if (FX1_WhoAmI(&who)!=ERR_OK) {
```

```
                return ERR_FAILED;
            }
                        len = sprintf(write, "Who Am I  value in
decimal \t: %4d\n",who);

                        SM1_SendBlock(SM1_DeviceData, &write, len);
            for(delay = 0; delay < 300000; delay++); //delay

            // Set up registers for accelerometer and magnetometer
values
            if (FX1_WriteReg8(FX1_CTRL_REG_1, 0x00) != ERR_OK) {
                return ERR_FAILED;
            }
            if (FX1_WriteReg8(FX1_M_CTRL_REG_1, 0x1F) != ERR_OK) {
                return ERR_FAILED;
            }
            if (FX1_WriteReg8(FX1_M_CTRL_REG_2, 0x20) != ERR_OK) {
                return ERR_FAILED;
            }
            if (FX1_WriteReg8(FX1_XYZ_DATA_CFG, 0x00) != ERR_OK) {
                return ERR_FAILED;
            }
            if (FX1_WriteReg8(FX1_CTRL_REG_1, 0x0D) != ERR_OK) {
                return ERR_FAILED;
            }

                        for(delay = 0; delay < 300000; delay++);
            // Get the X Y Z magnetometer values
            if (FX1_GetMagX(&magX)!=ERR_OK) {
                return ERR_OK;
            }
            if (FX1_GetMagY(&magY)!=ERR_OK) {
                return ERR_OK;
            }
            if (FX1_GetMagZ(&magZ)!=ERR_OK) {
                return ERR_OK;
            }
            printf("Magnetometer value \tX: %4d\t Y: %4d\t Z: %4d\n",
```

```
magX, magY, magZ);
            len = sprintf(write, "Magnetometer value \tX: %4d\t Y:
%4d\t Z: %4d\n", magX, magY, magZ);
          SM1_SendBlock(SM1_DeviceData, &write, len);
            for(delay = 0; delay < 300000; delay++); //delay
      }
```

Arduino IDE Program (Arduino Uno Code):

```
// include the library code:
#include <LiquidCrystal.h>
#include <SPI.h>
#include <Adafruit_GPS.h> // Initialize GPS
Adafruit_GPS GPS(&Serial);


char buff [255];
volatile byte indx;
volatile boolean process;
// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 7, en = 6, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
const int buzzer = 9; //buzzer to arduino pin 9
void setup() {
   Serial.begin (115200);
Serial1.begin(115200); // Debugging serial
GPS.begin(9600); // GPS baud rate
GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); // 1 Hz update rate delay(1000);

   pinMode(MISO, OUTPUT); // have to send on master in so it set as output
   SPCR |= _BV(SPE); // turn on SPI in slave mode
   indx = 0; // buffer empty
   process = false;
   SPI.attachInterrupt(); // turn on interrupt
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
```

```
  pinMode(buzzer, OUTPUT);
  pinMode(8, OUTPUT);
    pinMode(10, OUTPUT);
}

void turnOnGreen(){
  digitalWrite(8,HIGH);
}

ISR (SPI_STC_vect) // SPI interrupt routine
{
   byte c = SPDR; // read byte from SPI Data Register

   if (indx < sizeof(buff)) {
      buff[indx++] = c; // save data in the next index in the array buff
      if (c == '\n') {
        buff[indx - 1] = 0; // replace newline ('\n') with end of string (0)
        process = true;
      }
   }
}
int value;
void loop() {

 if (process) {
      process = false; //reset the process
     if(buff[0] == 'M'){
       lcd.clear();
       noTone(buzzer);
digitalWrite(8,0);
      Serial.print (buff[23]); //print the array on serial monitor
      Serial.print(buff[24]); //print the array on serial monitor
      Serial.print (buff[25]); //print the array on serial monitor
      Serial.println(buff[26]); //print the array on serial monitor
    if(buff[23]=='-'){
    value = buff[24] - '0';
    value = value * 100;
    value = value + (buff[25] - '0') * 10;
```

```
  value = value + (buff[26] - '0');
  value = value * -1;
  }
  else{
      value = buff[24] - '0';
  value = value * 100;
  value = value + (buff[25] - '0') * 10;
  value = value + (buff[26] - '0');
  }
      Serial.print("This is the value: ");
     Serial.println( value); //print the array on serial monitor
     Serial.println(buff);
    }
    indx= 0; //reset button to zero
if(value < 200 && value > 100){
  lcd.print("Boat is Stable");
 // turnOnGreen();
}
else if(value < -200 && value > -400){
  lcd.print("Tilting Right!!");
     // Serial.println("right");

 digitalWrite(8,1);

}
else if(value < -400){
  lcd.print("Ship Sinking");

   for(int i = 0; i <10; i++ ){
  digitalWrite(8,1);
  delay(20);
  digitalWrite(8,0);
  if(i ==4){
    tone(buzzer,1000);
  }
  }
}
else if(value < 100 && value > -200 && value != 0){
```

```
    lcd.print("Tilting Left!!");
        Serial.println(value);

   digitalWrite(8,1);
   // Serial.println("left");
  }
  else if(value < -200){
    lcd.print("Ship Sinking");

    for(int i = 0; i <10; i++ ){
    digitalWrite(8,1);
    delay(20);
    digitalWrite(8,0);
      if(i ==4){
      tone(buzzer,1000);
    }
    }
  }
     value = 0;

// GPS data display
if (GPS.fix) {
lcd.setCursor(0, 1);
lcd.print("Lat:");
lcd.print(GPS.latitude, 4);
lcd.setCursor(8, 1);
lcd.print("Lon:");
lcd.print(GPS.longitude, 4); }
 }
}
```

Part 4. Testing & Evaluation

Parts Required for Testing

- Protractor for Inclinometer (for verifying precise tilt measurements)
- Wave Generator (to simulate sea conditions)
- Calibration Weights (for precise adjustments)
- Adafruit Ultimate GPS Breakout (for GPS data verification)

Introduction: This section discusses the testing environment and the calibration methodology used for the Maritime Safety and Stability Alert System (MSSAS). The MSSAS is designed to provide visual and audio warnings in response to excessive tilting of a boat. The testing environment simulated real-world maritime conditions to ensure the system's reliability, accuracy, and additional functionality of GPS data integration.

Testing Environment

The testing environment aimed to replicate real-world maritime conditions, focusing on scenarios where boat tilting could occur due to weather conditions or control malfunctions. The following setups were utilized:

- Wave Generator: Used to simulate varying sea conditions, including calm seas and rough waters, to test the MSSAS under different levels of boat tilt.
- Calibration Weights: Applied to the boat model to adjust and control the tilt angle accurately for testing purposes.
- GPS Integration: The Adafruit Ultimate GPS Breakout was included to verify the system's ability to provide accurate location data alongside tilt warnings.

Calibration Methodology

To ensure the accurate and reliable operation of the MSSAS, a thorough calibration process was conducted, involving the following steps:

- Sensor Placement: The K64F microcontroller with its magnetometer was strategically placed within the boat model to capture representative tilt data.
- Data Collection: The magnetometer was connected to the data acquisition system (the Arduino Uno), which recorded tilt angles at regular intervals.
- Baseline Measurements: Baseline tilt measurements were recorded with the boat in a stable, upright position. These readings served as a reference point for subsequent experiments.

- Simulated Conditions: The wave generator was used to simulate various sea conditions. The tilt readings were continuously monitored to observe the MSSAS response.
- GPS Calibration: The GPS module was verified for accurate location data. Initial GPS coordinates were recorded and compared to known positions to ensure correct readings.
- Tilt Angle Calibration: Based on the results, calibration adjustments were made to the MSSAS to ensure accurate tilt detection and appropriate warning triggers.

Testing Scenarios

Scenario 1 - Testing Different Tilt Angles by Adjusting the Boat's Position:

Objective: To assess the MSSAS's response to different tilt angles and ensure accurate warning triggers.

Procedure:

1. Set the boat in a stable, upright position and record the baseline tilt angle.
2. Incrementally tilt the boat to various angles (e.g., 10°, 20°, 30°) using calibration weights or manual adjustments.
3. Observe and record the MSSAS's visual and audio warnings for each tilt angle.
4. Repeat steps 2-3 for both left and right tilts to ensure consistent performance.
5. Verify that the system provides accurate and timely warnings at the specified tilt thresholds.

Scenario 2 - Testing the System's Response to Simulated Rough Sea Conditions:

Objective: To evaluate the MSSAS's ability to respond to sudden and fluctuating tilts caused by simulated rough sea conditions.

Procedure:

1. Ensure the MSSAS is operating with baseline measurements recorded.
2. Activate the wave generator to simulate rough sea conditions with varying wave heights and frequencies.
3. Observe the MSSAS's response to the fluctuating tilts in terms of visual and audio warnings.
4. Record the system's performance and ensure it provides timely and accurate alerts for significant tilts.

Scenario 3 - Manually Triggering the System to Validate Warning Mechanisms:

Objective: To test the manual triggering capability of the MSSAS's warning mechanisms.

Procedure:

1. Set the MSSAS to manual mode, where specific tilt angles can be simulated manually.
2. Manually adjust the boat to simulate tilt angles beyond the safe threshold (e.g., more than 20°).
3. Observe the visual and audio warnings triggered by the MSSAS.
4. Repeat the manual adjustments for both left and right tilts to validate consistent warning triggers.

Scenario 4 - GPS Data Verification:

Objective: To ensure the MSSAS accurately provides GPS data along with tilt warnings. It was necessary to verify that GPS data was continuously gathered with accurate coordinates at all time in case of an emergency.

Procedure:

1. Verify initial GPS coordinates with known positions.
2. Move the boat model to various locations within the test environment.
3. Observe and record GPS data displayed on the LCD.
4. Ensure GPS data updates in real-time and accurately reflects the boat's position.

Conclusion: The four test cases, including testing different tilt angles, testing the system's response to simulated rough sea conditions, and manually triggering the system to validate warning mechanisms, and verifying GPS data integration, provide valuable insights into the performance and functionality of the MSSAS. These tests evaluate the system's responsiveness, accuracy, functionality of GPS data, and reliability in real-world maritime conditions. This verifies its effectiveness in enhancing boat safety and stability.

Part 5. Discussion

Limitations & Advances Introduction

This section will discuss the limitations encountered in our MSSAS project and potential solutions to address these limitations in future iterations.

Microcontroller Choice
The K64F microcontroller, while capable, presents certain limitations compared to more advanced microcontrollers available today. For instance, microcontrollers like the Raspberry Pi and ESP32 offer enhanced capabilities such as built-in WiFi, Bluetooth, and more robust data handling. These features enable faster response times and more efficient data communication, reducing interference and simplifying the overall configuration process. The K64F's

complexities, such as extensive pin configuration and limited data transmission capabilities, can hinder project development. Therefore, transitioning to a more advanced microcontroller could significantly enhance the system's performance and user experience.

Data Transmission Issues
During the development of the MSSAS project, we encountered issues with reliable data transmission. The system occasionally displayed random symbols and sometimes experienced crashes when sending sensor data. Despite extensive troubleshooting, the problem persisted intermittently, likely due to noise interference. Implementing more robust data transmission protocols or using microcontrollers with integrated communication modules could mitigate these issues. Ensuring proper shielding and grounding practices in the circuit design could also reduce noise and improve data integrity.

Power Constraints
A significant challenge with the K64F is its 3.3V operating voltage, whereas most peripherals and sensors typically operate at 5V. This voltage discrepancy necessitated careful design considerations to prevent over-voltage damage to the K64F. This limitation was particularly problematic given the project’s need for various sensors and communication modules, which often operate at 5V. Future designs could benefit from microcontrollers that natively support 5V logic levels, such as the Arduino or ESP32, to streamline integration and reduce the risk of power-related issues.

GPS and Water Sensor Integration
The addition of the Adafruit Ultimate GPS Breakout and a water level sensor introduced new complexities. Ensuring accurate GPS data and reliable water level detection required thorough calibration and testing. The GPS module occasionally struggled with acquiring signals in dense environments, and the water sensor's responsiveness varied based on water salinity and temperature. Future versions should consider more advanced GPS modules with better signal acquisition capabilities and water sensors that are more consistent across varying conditions.

Sensor Performance
The performance of the sensors integrated with the K64F demonstrated some inadequacies, affecting the reliability of the MSSAS system. To address this, we integrated more reliable sensors that provided accurate readings. Future iterations should focus on ensuring that all sensors are thoroughly tested and validated for their intended application, possibly considering alternative sensors known for higher accuracy and reliability.

In short, the limitations encountered during this project highlight the importance of selecting appropriate hardware and ensuring robust data transmission and power management. While the K64F microcontroller served as a functional prototype, transitioning to more advanced

microcontrollers like the ESP32 or Raspberry Pi could significantly enhance system performance and user experience. Addressing these limitations through hardware upgrades, improved communication protocols, and thorough testing will pave the way for a more reliable and efficient Maritime Safety and Stability Alert System in future developments.

Challenges & Drawbacks

This section will discuss the challenges and drawbacks encountered during the development and implementation of our MSSAS project.

Realistic Challenges

Display and Data Transmission Issues:
One of the primary challenges was reliably transmitting and displaying sensor data. Initially, we struggled with data transmission consistency, often seeing random symbols or system crashes. After several iterations of modifying the code and troubleshooting, we achieved more reliable performance. However, integrating the system components presented new challenges. The serial connection responsible for data reading and transmission conflicted with other system operations, causing data display malfunctions and incorrect data readings. Troubleshooting revealed noise interference as a significant issue, leading us to focus on ensuring accurate data output.

Voltage Compatibility and Noise:
Voltage requirements and noise were central challenges throughout the project. Most components operated at 5V, conflicting with the K64F's 3.3V logic. This discrepancy necessitated additional components, such as voltage level shifters, which introduced further complications. The K64F's data readings were unreliable due to noise and insufficient voltage levels. Isolating critical components improved performance but added complexity. The noise issues impacted serial communication and overall system reliability.

Drawbacks

K64F Microcontroller Limitations:
The K64F microcontroller posed several drawbacks. It is bulky, time-consuming to program, and costly. Its complexity added unnecessary steps to our project without providing significant benefits. In a commercial context, the K64F's high cost would be a major drawback, making it impractical for budget-conscious consumers. Cheaper and more versatile alternatives, such as custom IC chips or more modern microcontrollers, would be better suited for this application.

GPS and Water Sensor Limitations:

The GPS module faced difficulties in acquiring satellite signals in dense or obstructed environments, limiting its effectiveness in certain scenarios. Additionally, the water sensor's performance varied with different water conditions, such as changes in salinity or temperature, which affected its accuracy. Future designs should explore more reliable GPS solutions and water sensors that maintain consistent performance across diverse conditions.

Improvements and the Future

Enhanced Data Display and Control:
Implementing an app or web server for data display would improve user interaction, allowing for real-time monitoring and control from any location. Replacing the K64F with a more flexible and cost-effective microcontroller like the ESP8266 would streamline the system and reduce costs. Using sensors that accurately read data without noise interference would simplify integration and improve reliability.

Hardware and Circuit Improvements:
Designing a PCB to isolate specific sections and minimize noise would enhance system stability and performance. Upgrading to a robust external power supply would improve efficiency and functionality. Implementing better switching mechanisms, such as using MOSFETs or SSRs, would decrease noise and enhance consistency and speed.

GPS and Water Sensor Enhancements:
Future iterations should consider GPS modules with better signal acquisition capabilities and more consistent water sensors. Additionally, incorporating advanced algorithms for data filtering and error correction can enhance the reliability of GPS data and water level readings.

Final Product Goals

Personalized Settings:
Create tailored monitoring settings via an intuitive app interface, allowing users to adjust preferences easily from any location. This will have options including setting which environment conditions to look out for, what kind of alert the boat crew needs (Visual & Auditory), and other features that may help better alert the boat’s crew.

Energy Efficiency:
Incorporate advanced sensors to optimize energy consumption by adjusting monitoring and alert systems according to environmental conditions, promoting eco-conscious use. If energy consumption is not adequate or optimized, these systems may fail in a real world scenario.

Seamless Integration:
Ensure seamless integration with popular vehicles, allowing users to automate the MSSAS for many applications such as airplanes, seaships and automobiles.

Remote Access and Intelligent Modes:
Provide remote access, enabling users to control the system, adjust settings, and receive notifications regarding maritime conditions from anywhere. This would allow the user to turn off alerts without having to locate the actual system and flick off a switch. Implement intelligent modes that enhance safety and convenience by adjusting settings based on automatically recognized environmental conditions and user-defined criteria.

The final product aims to meet these goals, offering a user-friendly, and efficient Maritime Safety and Stability Alert System that enhances safety and convenience for users.

Part 6. Roles and Assignment Distributions

Johan Gonzales:

- Contributed to circuit assembly making sure serial communication between K64F and Arduino Uno was adequate.
  Made sure all components such as LCD, buzzer, LEDs were exhibiting desired behavior.
- Built project base/enclosure.
- Contributed to calibration of GPS coordinate values and made sure data was accurate

Cristian Molano:

- Led the circuit design making sure all components worked together correctly. Also implemented power supply connections and tested voltage levels.
- Programmed the K64F MCU to set up UART communication with processor expert, debugged data transmission component and set up magnetometer and accelerometer.
- Contributed to calibration of GPS coordinate values and made sure data was accurate

Martin Mejia:

- Set up a program for Arduino including the logic for different states, processing the data communicated from K64F, and ensuring the I/O components for Arduino work properly by debugging and case testing.
- Contributed to calibration of GPS coordinate values and made sure data was accurate
- Aided in K64F programming to verify magnetometer and accelerometer functionality.

Responsibilities were distributed according to every team member's strengths. Cristian showed strong programming skills with the K64F microcontroller hence why he led the programming for it. He also set up and got all libraries and code ready for the GPS module. Johan showed strong programming skills with Arduino hence why he led the programming for the Arduino Uno microcontroller. Martin demonstrated strong programming skills with state machines, which is why he was in charge of setting up all the different states in stability and warnings. All team members had experience in circuit design so everyone contributed to wiring all components together.

Part. 7 Conclusion

This project provided an opportunity for the team to deepen their understanding of the K64F microcontroller and enhance their circuit design and programming skills. At first, the project appeared straightforward, but as we delved deeper, it became evident that it would require significant time and effort. Our initial plan was to use the K64F's capabilities to monitor maritime conditions and alert users to potential safety issues. However, complications arose when we encountered issues with sensor integration and noise management, leading us to adapt our approach and implement additional solutions.

Our primary goal was to build a prototype Maritime Safety and Stability Alert System (MSSAS) that could reliably monitor maritime conditions and provide timely alerts. The significance of this project lies in enhancing maritime safety through advanced monitoring and alert systems, potentially preventing accidents and ensuring better preparedness for emergency situations.

Our methodology began with evaluating the available components and researching current sensor technologies. The initial design included a K64F microcontroller, various sensors for monitoring maritime conditions, and an Arduino for additional data processing and display purposes. However, sensor integration issues forced us to adopt alternative sensors and additional components, adding complexity to the system. Through extensive testing and adjustments, including noise reduction techniques and improved data transmission protocols, we managed to achieve reliable system performance.

The two main deliverables that were chosen to implement into this project was serial communication and use of GPIO pins. As previously mentioned the purpose of this project was to create a robust system that can be implemented into a variety of vehicles. It was known that the K64F contained a magnetometer and accelerometer, so it was decided to leverage this component to sample the y-axis of the boat, essentially measuring the tilt angles. Once these angles are calculated and processed, UART communication is used to transmit the data to a separate Arduino Uno microcontroller in the form of a character array.

In addition to tilt measurement, we integrated the Adafruit Ultimate GPS Breakout module into the system to provide real-time location data. This GPS module is crucial for maritime safety, enabling the MSSAS to send precise location coordinates in the event of an emergency. The GPS data is transmitted along with tilt information to the Arduino Uno, where it is processed and displayed. This integration ensures that not only can the system alert users to dangerous tilting, but it can also provide exact location coordinates for rescue operations. To further enhance the system's capabilities, a water level sensor was added to detect potential flooding. This sensor monitors water ingress within the boat, and when water levels exceed a certain threshold, the MSSAS triggers an alarm, providing an early warning to prevent sinking.

Once the data is gathered by the Arduino Uno, the next step is to convert the char. array into an integer to represent the tilt angle. Using an integer variable also allows for the comparison of the tilt sample in a conditional statement, thus creating different states for the system. For instance, one state can represent stability, another can represent instability and the last can represent imminent danger. If the calculated tilt angle is past a specified threshold, the system will start flashing the LED and buzzer indicating that the boat is either losing stability or that danger is imminent.

During the final demo, we ensured our system's functionality was clearly demonstrated. We used indicators to show the system's status, LEDs to display critical alerts, and switches to test various conditions. After numerous calibrations and tests, we successfully recorded a demonstration video showing our project in action.

Lessons learned from this project include the importance of pre-testing components, collaborative integration of system parts, and starting early to manage unexpected issues. Engaging with our TA proved invaluable for troubleshooting and guidance. Future research will focus on exploring alternatives to the K64F and Arduino, seeking components with built-in WiFi for better data management, and designing a PCB to minimize noise and improve system aesthetics.

In conclusion, this project was challenging and stressful, particularly during debugging. However, it was also a rewarding and educational experience. We gained practical skills in system integration and problem-solving, and we hope to revisit this project in the future, potentially transforming it from a class assignment into a marketable product. The integration of GPS and water sensors has added significant value to the MSSAS, enhancing its functionality and making it a more comprehensive maritime safety solution.

Demo Video Link:
https://drive.google.com/file/d/19WA4guCTIY9sTCzI0R3eYi-cAICVUyg6/view?usp=sharing