CORDIC modo rotador

Circuitos lógicos programables

Escrito por:

Karen Tatiana Zamudio

Revisión A

Nicolas Álvarez

Universidad de Buenos Aires

11 abril 2024

Historial de cambios

Cuadro 1: Registro de Revisiones

Revisión	Detalles de los cambios realizados	Fecha
0	Creación del documento	11 de abril 2024
1	Entrega	23 de abril 2024

Circuitos lógicos programables

Tabla de contenido

1. Introducción

En el contexto del diseño digital y la implementación de algoritmos, el presente documento aborda la creación y análisis de un módulo de rotación basado en el algoritmo CORDIC (Coordinate Rotation Digital Computer). El algoritmo CORDIC es ampliamente utilizado en sistemas digitales para calcular funciones trigonométricas y transformaciones de coordenadas de manera eficiente.

Este documento proporcionará una visión general del diseño del módulo cordic rotator, destacando su funcionamiento, estructura interna y su implementación en una FPGA (Field-Programmable Gate Array). Además, se presentarán los resultados de simulaciones y se analizará el uso de recursos de la FPGA.

1.1. Objetivos

Analizar y documentar el diseño, implementación y funcionamiento del módulo cordic rotator basado en el algoritmo CORDIC, así como evaluar su eficiencia y aplicabilidad en sistemas digitales.

1. Describir el algoritmo CORDIC y su aplicación en rotación de coordenadas: proporcionar una explicación detallada del algoritmo CORDIC y cómo se utiliza para realizar rotaciones de coordenadas cartesianas en el diseño del módulo cordic rotator.

2. Implementar el módulo cordic rotator en Verilog: crear una implementación en Verilog del módulo cordic rotator, asegurando que el código refleje correctamente la lógica del algoritmo CORDIC y que sea adecuado para su uso en sistemas digitales.

3. Realizar simulaciones del módulo y analizar su comportamiento: ejecutar simulaciones del módulo cordic rotator en un entorno de simulación, utilizando diferentes casos de prueba para evaluar su funcionamiento en una variedad de escenarios. Analizar los resultados de las simulaciones para verificar la precisión y eficacia del diseño.

4. Evaluar el uso de recursos de la FPGA: realizar un análisis exhaustivo del uso de recursos de la FPGA por parte del diseño del módulo cordic rotator, incluyendo el número de LUTs, FFs y otros recursos utilizados. Identificar áreas de mejora en términos de eficiencia y optimización del diseño.

1.2. Alcance

El alcance de este documento abarca la descripción detallada del diseño e implementación del módulo cordic rotator utilizando el algoritmo CORDIC en Verilog. Se incluirá una explicación exhaustiva del funcionamiento del algoritmo CORDIC y su aplicación en la rotación de coordenadas cartesianas. Además, se proporcionará el código Verilog del módulo cordic rotator junto con explicaciones detalladas de su estructura interna y su funcionamiento.

Se realizarán simulaciones del módulo en un entorno adecuado para evaluar su comportamiento en diversos escenarios, y se analizarán los resultados obtenidos. Asimismo, se llevará a cabo un análisis del uso de recursos de la FPGA por parte del diseño, con el objetivo de evaluar su eficiencia y escalabilidad en sistemas digitales.

2. ¿Qué es CORDIC?

CORDIC (Coordinate Rotation Digital Computer) es un algoritmo utilizado para calcular funciones trigonométricas, como seno, coseno, arcotangente, y para realizar rotaciones y transformaciones de coordenadas de manera eficiente en sistemas digitales. Fue desarrollado en la década de 1950 por Jack E. Volder en Bell Labs.

El algoritmo CORDIC se basa en una serie de rotaciones y desplazamientos sucesivos que permiten aproximar funciones trigonométricas y realizar operaciones de rotación con un bajo costo computacional. Una de las características clave del algoritmo CORDIC es su capacidad para realizar estas operaciones utilizando únicamente operaciones de suma, resta y desplazamiento bit a bit, lo que lo hace adecuado para implementaciones en hardware digital.

El algoritmo CORDIC es especialmente útil en aplicaciones donde se requieren cálculos trigonométricos rápidos y precisos, como en sistemas de procesamiento de señales, sistemas de comunicación digital, gráficos por computadora, entre otros. Además, es altamente versátil y puede adaptarse para calcular una amplia gama de funciones trigonométricas y realizar diversas operaciones geométricas.

2.1. CORDIC trigonométrico

El algoritmo CORDIC (Coordinate Rotation Digital Computer) Trigonométrico es un método iterativo utilizado para calcular funciones trigonométricas como seno (sin), coseno

(cos), arcoseno (arcsin) y arcocoseno (arc cos). Funciona mediante iteraciones de rotación y desplazamiento, donde cada iteración aproxima gradualmente el valor deseado de la función trigonométrica. El algoritmo se basa en la representación de un ángulo en coordenadas cartesianas (X, Y) y utiliza rotaciones sucesivas para acercarse al ángulo objetivo. A medida que las iteraciones continúan, el valor de salida converge hacia el valor correcto con una precisión determinada.

Descripción de las iteraciones de rotación y desplazamiento utilizadas en el alg En cada iteración del algoritmo CORDIC Trigonométrico, se realiza una rotación y un desplazamiento de las coordenadas (X, Y). La rotación se realiza mediante el ajuste de los ángulos de las coordenadas, lo que permite acercarse al ángulo objetivo. El desplazamiento se utiliza para ajustar las magnitudes de las coordenadas y garantizar que la convergencia del algoritmo sea adecuada. Estas iteraciones se repiten hasta que se alcanza la precisión deseada en el cálculo de la función trigonométrica.

Por ejemplo, para calcular el seno de un ángulo dado utilizando el algoritmo CORDIC Trigonométrico, se inicializan las coordenadas (X, Y) con valores apropiados y se realizan iteraciones de rotación y desplazamiento hasta que se alcanza la precisión deseada. El valor de Y al final de las iteraciones representa el valor aproximado del seno del ángulo dado. De manera similar, se pueden calcular el coseno, el arcoseno y el arcocoseno utilizando adaptaciones del algoritmo CORDIC Trigonométrico.

La precisión y la convergencia del algoritmo CORDIC Trigonométrico dependen del número de iteraciones realizadas y del número de bits utilizados para representar los valores de las coordenadas. En general, el algoritmo converge rápidamente y proporciona resultados precisos con un número suficiente de iteraciones. Sin embargo, la precisión puede verse afectada por la representación finita de los números en sistemas digitales y por la resolución de las operaciones de rotación y desplazamiento.

El algoritmo CORDIC trigonométrico se utiliza en una amplia gama de aplicaciones en sistemas digitales, incluyendo procesamiento de señales, comunicaciones, gráficos por computadora, sistemas de navegación, y más. Se utiliza para calcular funciones trigonométricas de manera eficiente y precisa en sistemas donde se requieren cálculos frecuentes de ángulos y funciones trigonométricas. Su implementación en hardware digital es especialmente adecuada debido a su simplicidad y eficiencia en términos de recursos computacionales.

2.2. CORDIC vectorial

El algoritmo CORDIC Vectorial utiliza un enfoque similar al CORDIC Trigonométrico, pero se enfoca en manipular vectores en lugar de solo ángulos. Permite realizar rotaciones y transformaciones de coordenadas en sistemas de coordenadas cartesianas o polares de manera eficiente.

En un sistema de coordenadas cartesianas, las operaciones del CORDIC Vectorial pueden utilizarse para rotar un vector en el plano XY. Por otro lado, en un sistema de coordenadas polares, el algoritmo puede utilizarse para convertir coordenadas cartesianas a polares y viceversa, así como para realizar rotaciones de vectores en el plano radial.

La eficiencia y precisión del CORDIC Vectorial lo hacen adecuado para una variedad de aplicaciones en sistemas digitales, como procesamiento de señales, modulación y demodulación en sistemas de comunicaciones, y transformaciones geométricas en gráficos por computadora. Además, su estructura iterativa permite su implementación eficiente en hardware digital, lo que lo convierte en una opción popular en sistemas embebidos y sistemas de tiempo real donde los recursos son limitados.

2.3. CORDIC logarítmico

El algoritmo CORDIC logarítmico utiliza una serie de iteraciones para aproximar funciones logarítmicas y exponenciales, así como operaciones relacionadas como la raíz cuadrada y el exponente. A través de un proceso iterativo, el algoritmo calcula las aproximaciones de estas funciones con una precisión deseada.

Una de las características clave del CORDIC logarítmico es su capacidad para calcular estas funciones de manera eficiente y precisa utilizando un enfoque iterativo y estructuras de datos simples. Esto lo hace adecuado para su implementación en sistemas digitales donde se requieren cálculos de funciones no trigonométricas con recursos limitados.

El CORDIC logarítmico tiene una amplia gama de aplicaciones en sistemas digitales, incluyendo procesamiento de señales, compresión de datos, criptografía y más. Se utiliza en situaciones donde se requiere el cálculo rápido y preciso de funciones logarítmicas y exponenciales, así como operaciones relacionadas como la raíz cuadrada y el exponente. Su eficiencia y precisión lo hacen valioso en una variedad de aplicaciones de sistemas digitales.

3. Implementación CORDIC rotator

En esta sección, se detalla la implementación del módulo CORDIC Rotator en Verilog. El módulo CORDIC Rotator es una implementación hardware del algoritmo CORDIC (Coordinate Rotation Digital Computer), diseñado para realizar rotaciones de vectores en el plano cartesiano. El algoritmo CORDIC se utiliza ampliamente en sistemas digitales para calcular funciones trigonométricas, transformaciones de coordenadas y otras operaciones matemáticas de manera eficiente. En esta sección, se explicará paso a paso la implementación del módulo CORDIC rotator, incluyendo la configuración de parámetros, la definición de la tabla de arco tangente, las iteraciones del algoritmo y la salida del módulo. Cada parte de la implementación se analizará en detalle, acompañada de su correspondiente código en Verilog.

3.1. Configuración de Tiempo y Declaración de Módulo

El código comienza con la configuración del `timescale` y la definición de parámetros utilizados en el módulo. El parámetro `c parameter` determina el ancho de bits de los datos de entrada y salida, mientras que `STG` se utiliza para definir el ancho de bits de los vectores `X` e `Y`. Luego, se declara el módulo `cordic rotator` con sus entradas (`clock`, `angle`, `Xin`, `Yin`) y salidas (`Xout`, `Yout`).

```
‘timescale 1ns/100ps

module cordic_rotator (
    input clock,
    input signed [31:0] angle,
    input signed [c_parameter - 1:0] Xin,
    input signed [c_parameter - 1:0] Yin,
    output signed [c_parameter:0] Xout,
    output signed [c_parameter:0] Yout
);

parameter c_parameter = 16;
localparam STG = c_parameter;
```

3.2. Declaración de la Tabla `atan_table`

La tabla `atan_table` contiene los valores de la función arco tangente utilizados en el algoritmo CORDIC. Estos valores están precalculados y se utilizan para realizar las operaciones de rotación en el módulo. La tabla está definida como un arreglo de 31 bits con 31 entradas, cada una representando un ángulo específico.

```
wire signed [31:0] atan_table [0:30];
assign atan_table[00] = 32'b00100000000000000000000000000000;
// Resto de asignaciones de la tabla atan_table...
```

3.3. Inicialización de registros y asignación de cuadrante

Se inicializan los registros `X`, `Y` y `Z`, que se utilizan para almacenar los datos en cada etapa del algoritmo CORDIC. Luego, se calcula el cuadrante en el que se encuentra el ángulo de rotación para determinar cómo se deben tratar los datos de entrada.

3.3.1. Cuadrante I

En el cuadrante I, el ángulo de rotación se encuentra en el rango de 0 a $\pi/2$. En este caso, no es necesario realizar ninguna modificación en los datos de entrada, ya que están en el rango adecuado para ser procesados por el algoritmo CORDIC. Por lo tanto, los valores de las coordenadas X e Y del vector de entrada (`Xin` y `Yin`) se asignan directamente a los registros `X[0]` y `Y[0]`, respectivamente. Además, el ángulo de rotación (`angle`) se asigna directamente al registro `Z[0]`.

```
case (quadrant)
    2'b00: begin // Cuadrante I
        X[0] <=  Xin;
        Y[0] <=  Yin;
        Z[0] <=  angle;
```

Este caso se activa cuando los bits de signo del ángulo son 0, indicando que el ángulo se encuentra en el primer cuadrante.

3.3.2. Cuadrante II

En el cuadrante II, el ángulo de rotación se encuentra en el rango de $\pi/2$ a π. Para ajustar los datos de entrada al rango adecuado para el algoritmo CORDIC, se realiza una pre-rotación del ángulo. En este caso, el ángulo se resta de π para asegurar que esté dentro del rango de -$\pi/2$ a 0. Los valores de las coordenadas X e Y del vector de entrada se asignan a los registros `X[0]` y `Y[0]`, respectivamente, de la misma manera que en el cuadrante I.

```
2'b01: begin // Cuadrante II
        X[0] <=     Xin;
        Y[0] <=     Yin;
        Z[0] <=  32'b10000000000000000000000000000000 - angle;
```

Este caso se activa cuando el bit de signo más significativo del ángulo es 0 y el siguiente bit es 1, indicando que el ángulo se encuentra en el segundo cuadrante.

3.3.3. Cuadrante III

En el cuadrante III, el ángulo de rotación se encuentra en el rango de -π a -$\pi/2$. Al igual que en el cuadrante II, se realiza una pre-rotación del ángulo restando π para asegurar que esté dentro del rango de -$\pi/2$ a 0. Los valores de las coordenadas X e Y del vector de entrada se asignan a los registros `X[0]` y `Y[0]`, respectivamente, de la misma manera que en los cuadrantes I y II.

```
2'b10: begin // Cuadrante III
        X[0] <=     Xin;
        Y[0] <=     Yin;
        Z[0] <=     angle - 32'b10000000000000000000000000000000;
```

Este caso se activa cuando el bit de signo más significativo del ángulo es 1 y el siguiente bit es 0, indicando que el ángulo se encuentra en el tercer cuadrante.

3.3.4. Cuadrante IV

En el cuadrante IV, el ángulo de rotación se encuentra en el rango de -$\pi/2$ a 0. Para ajustar los datos de entrada al rango adecuado para el algoritmo CORDIC, se realiza una pre-rotación del ángulo restando 2π para asegurar que esté dentro del rango de -$\pi/2$ a -π. Los valores de las coordenadas X e Y del vector de entrada se asignan a los registros `X[0]` y `Y[0]`, respectivamente, de la misma manera que en los cuadrantes I, II y III.

```
2'b10: begin // Cuadrante III
        X[0] <=     Xin;
        Y[0] <=     Yin;
        Z[0] <=     angle - 32'b10000000000000000000000000000000;
```

Este caso se activa cuando los bits de signo del ángulo son 1, indicando que el ángulo se encuentra en el cuarto cuadrante.

3.4. Banco de pruebas

El banco de pruebas proporcionado se encarga de simular el módulo `cordic_rotator` y generar formas de onda de entrada para probar su funcionalidad. Aquí tienes una explicación detallada de su funcionamiento:

3.4.1. Parámetros y variables locales

Se definen algunos parámetros y variables locales para el tamaño de los datos de entrada (`SZ`), los valores booleanos (`FALSE` y `TRUE`), y el valor de la señal de entrada (`VALUE`), que se reduce para ajustarse a la ganancia del sistema.

```
localparam SZ = 16;

reg [SZ-1:0] Xin, Yin;
reg [31:0] angle;
wire [SZ:0] Xout, Yout;
reg CLK_100MHZ;

localparam FALSE = 1'b0;
localparam TRUE = 1'b1;

// reducido por un factor de 1.647 ya que es la ganancia del sistema (k)
localparam VALUE = 32000/1.647;
```

3.4.2. Inicialización

Antes de comenzar la simulación, se inicializan las variables y las señales de entrada. Se establece `start` en `FALSE`, se inicializa la señal de reloj `CLK_100MHZ`, el ángulo en cero, y se establece un valor inicial para `Xin` y `Yin`.

```
initial
begin
    start = FALSE;
    CLK_100MHZ = 1'b0;
    angle = 0;
    Xin = VALUE;
    Yin = 1'd0;
    i = 0;

    #1000;
    @(posedge CLK_100MHZ);
    start = TRUE;
```

3.4.3. Generación de formas de onda

Durante la simulación, se genera una señal de reloj de 100 MHz (`CLK_100MHZ`). Después de un breve período de espera, se activa `start` para indicar que la simulación comienza y se comienza a generar una secuencia de ángulos para probar el módulo `cordic_rotator`.

```
for (i = 0; i < 360; i = i + 1)

begin
    @(posedge CLK_100MHZ);
    #11700
    start = FALSE;
    angle = ((1 << 32)*(i+1))/360;
end
```

3.4.4. Generación del reloj

Se genera una señal de reloj de 100 MHz (`CLK_100MHZ`) con una duración de tiempo determinada por `CLK100_SPEED`.

```
parameter CLK100_SPEED = 10; // 100Mhz = 10nS


initial
begin
    CLK_100MHZ = 1'b0;
    $display("CLK_100MHZ_iniciado");
    #5;
    forever
    begin
        #(CLK100_SPEED/2) CLK_100MHZ = 1'b1;
        #(CLK100_SPEED/2) CLK_100MHZ = 1'b0;
    end
end
```

3.4.5. Salida del módulo

Finalmente, se asignan las salidas `Xout` y `Yout` del módulo con los valores calculados después de completar todas las iteraciones del algoritmo CORDIC. Estas salidas representan las coordenadas rotadas del vector de entrada.

```
        assign Xout = X[STG - 1];
        assign Yout = Y[STG - 1];
```

3.4.6. Identificación de los estados del sistema

El sistema Realità, al ser un sistema autónomo para la captura de información topográfica, presenta diversos estados durante su funcionamiento. Para una mejor comprensión del sistema, se detallan a continuación los estados principales:

1. Estado de inicio:

    a) El sistema se encuentra inactivo, a la espera de instrucciones.

b) Los sensores están desactivados y no se está capturando información.

c) El sistema espera la interacción del usuario para iniciar el proceso de captura.

2. Estado de configuración:

a) Se configuran los parámetros de operación de cada sensor, como la frecuencia de muestreo, la resolución y el rango de medición, de acuerdo con las características del área de captura y los requerimientos del usuario.

b) El sistema valida los parámetros ingresados y se prepara para la captura.

3. Estado de captura:

a) Se realiza la calibración interna de los sensores para garantizar la precisión y confiabilidad de las mediciones.

b) Los sensores comienzan a capturar datos de forma simultánea, registrando mediciones de su respectiva naturaleza.

4. Estado de procesamiento:

a) El sistema fusiona los datos de los diferentes sensores en tiempo real.

b) Los datos capturados se procesan y analizan para generar un mapa topográfico preciso.

c) Se identifican y clasifican los elementos del entorno, como edificios, calles y vegetación.

d) Se generan metadatos que describen el mapa y las condiciones de captura.

5. Estado de finalización:

a) La captura de datos y el procesamiento finalizan.

b) El sistema presenta al usuario el mapa topográfico generado, junto con los metadatos correspondientes.

c) El usuario puede guardar el mapa y los metadatos para su posterior uso.

6. Estado de error:

a) El sistema entra en este estado si se produce un error durante la captura, el procesamiento o la finalización del proceso.

b) Se registra el tipo de error y la causa probable.

c) El usuario es notificado sobre el error y se le proporcionan instrucciones para resolverlo.

3.5. Definición de los eventos

En el sistema Realità, los eventos que desencadenan las transiciones entre los diferentes estados son acciones o situaciones específicas que determinan el cambio de estado del sistema. Estos eventos pueden ser generados por el usuario, por los sensores o por el propio sistema como resultado de su funcionamiento.

A continuación, se detalla la lista de eventos que desencadenan cada estado:

1. Estado de inicio:

   a) Evento de inicio: el usuario enciende el sistema o inicia una nueva sesión de captura.

2. Estado de configuración:

   a) Evento de inicio de configuración: el usuario accede al menú de configuración del sistema.

   b) Evento de modificación de parámetros: el usuario modifica los parámetros de captura, como el área de interés, la resolución deseada o el tipo de sensores a utilizar.

   c) Evento de validación de parámetros: el sistema valida los parámetros ingresados por el usuario.

   d) Evento de preparación para la captura: el sistema se prepara para iniciar la captura de datos de acuerdo con los parámetros configurados.

3. Estado de captura:

   a) Evento de inicio de captura: el usuario da la orden de iniciar la captura de datos.

   b) Evento de activación de sensores: los sensores se activan y comienzan a capturar datos.

c) Evento de calibración interna: se realiza la calibración interna de los sensores para garantizar la precisión y confiabilidad de las mediciones.

d) Evento de captura de datos: los sensores capturan datos de forma simultánea, registrando mediciones de su respectiva naturaleza.

e) Evento de fusión de datos en tiempo real: el sistema fusiona los datos de los diferentes sensores en tiempo real.

4. Estado de procesamiento:

a) Evento de finalización de captura: se alcanza el área de captura definida o se agota el tiempo de captura.

b) Evento de inicio de procesamiento: los datos capturados están listos para ser procesados.

c) Evento de procesamiento de datos: el sistema procesa y analiza los datos capturados para generar un mapa topográfico preciso.

d) Evento de identificación y clasificación de elementos: se identifican y clasifican los elementos del entorno, como edificios, calles y vegetación.

e) Evento de generación de metadatos: se generan metadatos que describen el mapa y las condiciones de captura.

5. Estado de finalización:

a) Evento de finalización de procesamiento: el procesamiento de datos finaliza y el mapa topográfico está generado.

b) Evento de presentación del mapa: el sistema presenta al usuario el mapa topográfico generado, junto con los metadatos correspondientes.

c) Evento de guardado del mapa: el usuario guarda el mapa y los metadatos para su posterior uso.

6. Estado de error:

a) Evento de detección de error: se produce un error durante la captura, el procesamiento o la finalización del proceso.

b) Evento de registro de error: se registra el tipo de error y la causa probable.

c) Evento de notificación de error: el usuario es notificado sobre el error y se le proporcionan instrucciones para resolverlo.

7. Consideraciones adicionales:

a) En algunos casos, un evento puede desencadenar la transición a más de un estado. Por ejemplo, el evento de detección de error puede desencadenar la transición al estado de error y al estado de finalización, dependiendo de la gravedad del error.

b) El sistema puede incluir mecanismos de recuperación de errores para intentar resolver los errores y volver al estado anterior.

c) La secuencia de eventos puede variar dependiendo de la configuración del sistema y las condiciones de funcionamiento.

3.6. Componer el árbol de transiciones

El árbol de transiciones es una representación visual que muestra todas las posibles transiciones entre los estados del sistema y los eventos que las desencadenan. Esta representación visual es útil para comprender mejor la estructura del sistema y diseñar casos de prueba efectivos.

Para componer el árbol de transiciones del sistema Realità, primero se identifican todos los estados y eventos. Luego, se trazan las transiciones entre ellos de acuerdo con las reglas definidas. Una vez completado el árbol de transiciones, se puede visualizar de manera clara cómo interactúan los diferentes elementos del sistema en respuesta a diversos eventos.

3.7. Creación de casos de prueba legales

Los casos de prueba legales son aquellos que verifican el comportamiento correcto del sistema cuando se presentan eventos válidos. Estos casos de prueba están diseñados para probar todas las transiciones permitidas entre estados y asegurarse de que el sistema responda correctamente a cada evento.

Para crear los casos de prueba legales para el sistema Realità, se identifican todos los eventos válidos y se diseñan casos de prueba específicos para cada uno de ellos. Estos casos de prueba deben cubrir todas las transiciones entre estados permitidas por el sistema, verificando que el sistema se comporte según lo esperado en cada situación.

Cuadro 2: Tabla de estados y eventos del sistema

Estado	**Evento**	**Siguiente Estado**	**Descripción**
Inicio	Inicio del sistema	Configuración	El usuario enciende el sistema o inicia una nueva sesión de captura.
	Modificación de parámetros	Configuración	El usuario modifica los parámetros de captura.
	Validación de parámetros	Configuración	El sistema valida los parámetros ingresados por el usuario.
Configuración	Preparación para la captura	Captura	El sistema se prepara para iniciar la captura de datos.
Captura	Inicio de captura	Captura	El usuario da la orden de iniciar la captura de datos.
	Activación de sensores	Captura	Los sensores se activan y comienzan a capturar datos.
	Calibración interna	Captura	Se realiza la calibración interna de los sensores.
	Captura de datos	Captura	Los sensores capturan datos simultáneamente.
	Fusión de datos en tiempo real	Captura	El sistema fusiona los datos de los sensores.
	Finalización de captura	Procesamiento	Se alcanza el área de captura definida o se agota el tiempo de captura.
Procesamiento	Inicio de procesamiento	Procesamiento	Los datos capturados están listos para ser procesados.
	Procesamiento de datos	Procesamiento	El sistema procesa y analiza los datos capturados.
	Identificación y clasificación de elementos	Procesamiento	Se identifican y clasifican los elementos del entorno.
	Generación de metadatos	Procesamiento	Se generan metadatos que describen el mapa y las condiciones de captura.
	Finalización de procesamiento	Finalización	El procesamiento de datos finaliza y el mapa topográfico está generado.
Finalización	Presentación del mapa	Finalización	El sistema presenta al usuario el mapa topográfico generado.
	Guardado del mapa	Finalización	El usuario guarda el mapa y los metadatos para su posterior uso.
Cualquiera	Detección de error	Error	Se produce un error durante el proceso.
	Registro de error	Error	Se registra el tipo de error y la causa probable.
	Notificación de error	Error	El usuario es notificado sobre el error y se le proporcionan instrucciones.

Cuadro 3: Casos de prueba para Realità

ID	Evento	Guardia	Acción	Estado Inicial	Resultado Esperado
1	Inicio del sistema	-	-	Inicio	Configuración
2	Modificación de parámetros	-	-	Configuración	Configuración
3	Validación de parámetros	-	-	Configuración	Preparación para la captura
4	Inicio de captura	-	-	Captura	Captura
5	Activación de sensores	Sensores activos	-	Captura	Captura
6	Calibración interna	Sensores calibrados	-	Captura	Captura
7	Captura de datos	Datos capturados	-	Captura	Captura
8	Fusión de datos en tiempo real	Datos fusionados	-	Captura	Procesamiento
9	Inicio de procesamiento	Procesamiento iniciado	-	Procesamiento	Procesamiento
10	Procesamiento de datos	Datos procesados	-	Procesamiento	Procesamiento
11	Identificación y clasificación de elementos	Elementos identificados	-	Procesamiento	Procesamiento
12	Generación de metadatos	Metadatos generados	-	Procesamiento	Procesamiento
13	Finalización de procesamiento	Procesamiento finalizado	-	Procesamiento	Finalización
14	Presentación del mapa	Mapa presentado	-	Finalización	Finalización
15	Guardado del mapa	Mapa guardado	-	Finalización	Finalización
16	Detección de error	Error detectado	-	Finalización	Error

3.8. Creación de casos de prueba ilegales

Los casos de prueba ilegales son aquellos que prueban el comportamiento del sistema cuando se presentan eventos inválidos o inesperados. Estos casos de prueba están diseñados para evaluar cómo el sistema maneja situaciones incorrectas y asegurarse de que responda de manera adecuada, como mostrar mensajes de error o tomar medidas correctivas.

Para crear los casos de prueba ilegales para el sistema Realità, se identifican eventos que podrían causar transiciones no permitidas entre estados o situaciones incoherentes en el sistema. Luego se diseñan casos de prueba específicos para estos eventos, verificando que el sistema maneje adecuadamente estas condiciones inesperadas.

Cuadro 4: Casos de prueba ilegales para Realità

No.	Configuración	Estado	Evento	Resultado
1	-	Inicio	Modificación de parámetros	Error
2	Configuración	Captura	Inicio de captura	Error
3	Captura	Procesamiento	Inicio de procesamiento	Error
4	Procesamiento	Finalización	Presentación del mapa	Error

3.9. Creación de casos de prueba para los guardias

Los casos de prueba para los guardias son aquellos diseñados para evaluar las condiciones de seguridad y protección del sistema. Estos casos de prueba se centran en verificar que el sistema Realità pueda detectar y responder adecuadamente a situaciones de seguridad, como intentos de acceso no autorizado o manipulación maliciosa de datos.

Para crear los casos de prueba para los guardias en el sistema Realità, se identifican posibles vulnerabilidades o amenazas de seguridad y se diseñan casos de prueba específicos para evaluar la capacidad del sistema para detectar y mitigar estas amenazas. Esto puede incluir pruebas de autenticación de usuarios, cifrado de datos, detección de intrusiones y otras medidas de seguridad.

Cuadro 5: Casos de prueba para la detección de errores

No.	**Estado Inicial**	**Evento**	**Estado Siguiente**	**Descripción del Caso de Prueba**
1	Inicio	Detección de error	Error	Verificar que el sistema maneje correctamente la detección de un error al iniciar.
2	Configuración	Registro de error	Error	Verificar que el sistema registre adecuadamente un error durante la configuración.
3	Captura	Detección de error	Error	Verificar que el sistema maneje correctamente la detección de un error durante la captura.
4	Procesamiento	Detección de error	Error	Verificar que el sistema maneje adecuadamente la detección de un error durante el procesamiento.
5	Finalización	Detección de error	Error	Verificar que el sistema maneje correctamente la detección de un error durante la finalización.

4. Bases del test

Esta sección proporciona una visión general del proyecto, incluyendo los siguientes aspectos:

4.1. Especificaciones del producto

4.1.1. Diseño funcional general

El diseño funcional general ofrece una descripción panorámica de las funciones clave del producto, abordando sus características y capacidades fundamentales.

4.1.2. Diseño funcional detallado

Este documento detallado presenta una visión a profundidad de las funciones específicas del producto, proporcionando información pormenorizada sobre su implementación y operación.

4.1.3. Guía del usuario Realità

La guía del usuario Realità se presenta como un manual exhaustivo destinado a los usuarios finales, contiene instrucciones detalladas para aprovechar al máximo las capacidades del producto, garantizando una experiencia de usuario óptima.

4.2. Normas

4.2.1. Normas internas para productos de prueba

Establecemos normas internas que actúan como directrices esenciales para la creación y ejecución de productos de prueba, asegurando coherencia y calidad en todo el proceso.

4.2.2. Libro Testing Embedded Software

Este libro representa una fuente de referencia esencial que aborda las mejores prácticas en pruebas de software integrado, su contenido se utiliza como guía para garantizar un enfoque sólido y eficiente en la fase de pruebas.

4.3. Manuales de usuario

4.3.1. Manual del entorno de prueba del usuario

Este manual ofrece instrucciones detalladas sobre el entorno de prueba, proporcionando a los usuarios información clara y completa para una interacción efectiva con el sistema.

4.3.2. Manual de herramientas de prueba del usuario

La guía sobre las herramientas de prueba ofrece información detallada sobre las herramientas utilizadas en el proceso de prueba, asegurando una comprensión profunda de su funcionamiento y aplicación.

4.4. Planes del proyecto

4.4.1. Plan de proyecto Realità

En el plan de proyecto Realità, se detallan todos los aspectos relacionados con la gestión y ejecución del proyecto, desde los objetivos hasta la asignación de recursos, proporciona una hoja de ruta integral para el éxito del proyecto.

4.5. Planificación

4.5.1. Planificación del equipo de desarrollo Realità

La planificación detallada del equipo de desarrollo para Realità establece la estructura temporal y las tareas asignadas, garantizando una ejecución eficiente y coordinada de las actividades del equipo durante todo el proyecto.

5. Estrategía general del test

La estrategia de pruebas de aceptación para Realità se basa en la premisa de que las pruebas de módulos y de integración son realizadas por el equipo de desarrollo. Esta estrategia se ha elaborado después de reuniones con el comisionado, líder del trabajo y gerente de pruebas específicamente para el desarrollo del trabajo Realità.

5.1. Características de calidad y su importancia relativa

Subcaracterísticas seleccionadas:

1. Idoneidad (Suitability): verificación de que las funcionalidades e interfaces cumplan con los requisitos establecidos.

2. Operabilidad (Operability): evaluación de la interfaz de usuario en términos de simplicidad e intuición.

3. Precisión (Accurateness): mantenimiento de la precisión de los valores medidos en diversas condiciones de funcionamiento.

4. Eficiencia (Time Behaviour): garantizar que el tiempo de respuesta ante eventos críticos sea aceptable.

5. Mantenibilidad (Changeability): facilitar la actualización y adaptación del software.

5.2. Matriz de estrategia para Realità

Realità, como sistema autónomo diseñado para la captura eficiente y precisa de información topográfica, despliega diversas funciones esenciales para cumplir con los objetivos del proyecto.

Subsistema	Funcionalidad	Usabilidad	Eficiencia	Mantenibilidad
SAD	++	++	+	
SP	++	+	++	+
SFD3D	++	++	++	++
SC	+	++	+	++
Importancia relativa (100)	40	20	25	15

Referencias

- ++ La característica de calidad es predominante para el subsistema.
- + La característica de calidad es relevante para el subsistema.
- (vacío) La característica de calidad es insignificante para el subsistema.

A continuación, se detallan los subsistemas funcionales clave de Realità en relación con la captura de datos topográficos:

1. **Subsistema adquisición de datos (SAD)**

   a) Realità emplea una variedad de sensores y controladores para garantizar una captura eficiente de datos topográficos:

   - Sensores LiDAR: utilizados para la detección remota y la generación de modelos tridimensionales precisos del entorno.
   - Sensores GPS: proporcionan información geoespacial precisa para la localización y el mapeo del terreno.
   - Sensores de imagen con profundidad: permiten la captura de imágenes tridimensionales del entorno con información de profundidad.
   - Sensores IMU (Unidad de Medición Inercial): utilizados para medir y registrar los movimientos y la orientación del dispositivo en tiempo real.
   - Controladores de movimiento: responsables de coordinar y controlar los movimientos del dispositivo para garantizar una captura precisa y eficiente de los datos topográficos.

   Estos componentes trabajan en conjunto para proporcionar una captura de datos integral y precisa, permitiendo a Realità cumplir con sus objetivos de manera efectiva.

2. **Subsistema de procesamiento de datos (SP)**

   a) **Microcontrolador (MC):** el microcontrolador STM32 integra toda la información de los sensores y coordina el procesamiento de datos para su posterior análisis y utilización.

3. **Subsistema de fusión de datos 3D (SFD3D)**

   a) **Generador de Nube de Puntos 3D:** este componente genera la nube de puntos 3D a partir de los datos fusionados, proporcionando una representación detallada del terreno topográfico.

   b) **Clasificación de Puntos 3D:** este algoritmo clasifica los datos capturados por los diferentes sensores para crear una representación tridimensional precisa del entorno.

4. **Subsistema de comunicación (SC)**

a) **Sensores Wi-Fi, Bluetooth y de radio:** estos sensores permiten la comunicación inalámbrica para la coordinación, manejo, inicio y finalización de la información entre Realità y otros dispositivos o estaciones base.

5. **Integración sinérgica del sistema completo**

a) **Integración holística para una captura integral:** la integración completa que proporciona una solución holística, maximizando la eficiencia y precisión en la captura de datos topográficos alineados con los requerimientos del proyecto.

5.3. Técnicas de diseño de pruebas

Las técnicas de diseño de pruebas desempeñan un papel crucial en el proceso de aseguramiento de la calidad de Realità, estas técnicas proporcionan el marco fundamental para evaluar la funcionalidad, usabilidad, confiabilidad y precisión del sistema autónomo de captura de información topográfica.

En el contexto específico de Realità, las técnicas de diseño de pruebas se adaptan de manera precisa y meticulosa para abordar los desafíos únicos asociados con la integración de sensores multimodales, como GPS, LIDAR, sensor de imagen e IMU.

5.3.1. Pruebas de funcionalidad

Descripción: verificar la funcionalidad de cada componente del sistema Realità, incluyendo la captura de datos, procesamiento y generación de mapas.

Técnicas:

1. Pruebas unitarias: evaluar cada módulo individualmente.

2. Pruebas de integración: verificar la interacción entre módulos, especialmente en la fusión de datos de múltiples sensores.

3. Pruebas de sistema: validar la funcionalidad del sistema en su conjunto.

5.3.2. Pruebas de usabilidad

Descripción: evaluar la facilidad de uso del sistema Realità para los usuarios finales, centrándose en la interpretación de los mapas generados.

Técnicas:

1. Pruebas de interfaz de usuario (UI): Evaluar la interfaz para asegurar una experiencia intuitiva.

2. Pruebas de navegación: verificar la facilidad de movimiento y acceso a funciones clave.

5.3.3. Pruebas de confiabilidad

Descripción: asegurarse de que Realità produce resultados precisos y consistentes bajo diversas condiciones.

Técnicas:

1. Pruebas de estabilidad: evaluar la capacidad del sistema para manejar cargas de trabajo sostenidas.

2. Pruebas de estrés: exponer el sistema a condiciones extremas para evaluar su comportamiento.

5.3.4. Pruebas de fusión de datos

Descripción: garantizar que la integración de datos de sensores GPS, LIDAR, sensor de imagen e IMU genere mapas detallados y precisos.

Técnicas:

1. Pruebas de fusión de datos 3D: evaluar la capacidad del sistema para generar representaciones tridimensionales precisas.

2. Pruebas de coherencia de datos: verificar que los datos de diferentes sensores se integren de manera coherente.

5.3.5. Pruebas de captura de datos en campo

Descripción: evaluar el rendimiento de Realità durante la captura de datos topográficos en situaciones del mundo real.

Técnicas:

1. Pruebas de campo simuladas: emular condiciones del mundo real para evaluar la precisión y eficacia de la captura de datos.

Prueba	Idoneidad	Operabilidad	Precisión	Tiempo	Cambiabilidad
Unitarias		++	+	+	+
Integración		++		++	
Sistema		+	++	++	
UI			++		
Navegación			++		
Estabilidad		+	++		
Estrés		+	++		
SFD3D	++	++	++	++	
Coherencia			++		
Simuladas		++	++	++	++

Cuadro 6: Matriz de Técnicas de Diseño para Pruebas en Realità

5.4. Esfuerzo estimado

La estimación de esfuerzo es un componente vital en la planificación de pruebas de aceptación para el proyecto Realità, al abordar la complejidad y amplitud del sistema autónomo de captura de datos topográficos, es esencial prever y asignar recursos de manera precisa.

Esta subsección se centra en la evaluación cuidadosa de las tareas relacionadas con las pruebas, considerando factores como la magnitud de la implementación, la diversidad de las técnicas de prueba y las posibles contingencias, una estimación realista y fundamentada no solo facilita la asignación adecuada de recursos, sino que también sirve como un indicador crucial para evaluar la viabilidad temporal y financiera del proyecto Realità.

5.5. Planificación

La planificación meticulosa es esencial para el éxito de cualquier proyecto, y las pruebas de aceptación para Realità no son la excepción, en esta fase, se delinea un camino claro que abarca desde la elaboración del plan de pruebas hasta la finalización del informe de resultados y conclusiones, cada actividad se ha calendarizado de manera estratégica para cumplir con el ambicioso objetivo de completar el proyecto dentro del periodo del 10 de marzo al 30 de junio.

Con un enfoque organizado y detallado, se busca asegurar la eficiencia y efectividad en cada etapa del proceso, contribuyendo así al desarrollo exitoso del sistema autónomo de

Actividad	**Esfuerzo (horas)**
Plan de pruebas	20
Fase de planificación y control	80
Gerencia de pruebas	48
Gerencia de configuración de pruebas	28
Soporte metodológico	44
Fase de preparación	32
Fase de especificación	80
Fase de ejecución	200
Fase de finalización	16
Total	**548**

Cuadro 7: Esfuerzo estimado para el proceso de pruebas

captura de datos topográficos.

6. Amenazas, riesgos y medidas

El entorno de pruebas para el proyecto Realità no está exento de desafíos potenciales y riesgos que podrían afectar el éxito de las pruebas de aceptación, por lo anterior se hace necesario identificar, evaluar y gestionar estas amenazas de manera proactiva garantizado la integridad y la eficacia del proceso de prueba.

En esta subsección, se abordarán posibles amenazas y riesgos, desde limitaciones tecnológicas hasta cambios inesperados en los requisitos, y se propondrán medidas mitigadoras específicas.

1. Posible retraso en la entrega a prueba: en caso de retrasos, se utilizará la estrategia de prueba para determinar las pruebas omitidas, asegurando una cobertura adecuada incluso bajo restricciones de tiempo.

2. Equipo de desarrollo disuelto después de la entrega: se garantizará la presencia de al menos un desarrollador experimentado durante la ejecución de pruebas para abordar cualquier problema inesperado y mantener el conocimiento del sistema.

3. Disponibilidad limitada del experto de dominio: se planificará de manera adecuada, anticipando posibles retrasos y asegurando la máxima disponibilidad del experto de

Actividad	Inicio	Fin	Responsable
Plan de pruebas	10/03	19/03	Equipo de Pruebas
Fase de Planificación			
Elaboración del documento de plan de pruebas	20/03	31/03	Líder de Pruebas
Revisión del plan con el equipo	01/04	09/04	Equipo de Pruebas
Configuración de pruebas	10/04	23/04	Ingeniero de Configuración
Soporte metodológico	24/04	07/05	Especialista en Metodología
Fase de Preparación			
Elaboración de casos de prueba	15/03	31/03	Analista de Pruebas
Revisión de casos de prueba	01/04	03/04	Equipo de Pruebas
Fase de Ejecución			
Ejecución de pruebas	10/05	26/05	Equipo de Pruebas
Fase de Finalización			
Informe de resultados y conclusiones	27/05	28/05	Líder de Pruebas

Cuadro 8: Planificación de Pruebas de Aceptación para *Realità*

dominio cuando sea necesario.

4. Diseñadores trabajando en el diseño funcional y guías de usuario: se garantizará la disponibilidad oportuna de los diseñadores para mantener la coherencia y reproducibilidad en el diseño funcional y las guías de usuario.

7. Infraestructura

7.1. Entorno de pruebas

El equipo de pruebas está compuesto por cinco personas, durante el proyecto de prueba, se debe disponer de dos (2) PC con una configuración de hardware estándar y el software estándar utilizado por Solidly Embedded Ltd. (para software especial, consulte la sección) que requiere un espacio mínimo de 20 GB en el directorio de gestión del producto.

7.2. Herramientas de pruebas

Las siguientes herramientas de prueba son esenciales:

1. Herramienta de gestión de defectos *DefectTracker*.
2. Herramienta de gestión de cambios *ChangeMaster*.
3. Software de planificación.
4. Hardware adicional para rastrear el comportamiento interno eferente a la captura de información topográfica.

7.3. Entorno

Se utiliza el entorno de prueba estándar, los probadores utilizan su espacio de oficina y campo durante el proyecto de prueba.

8. Organización de pruebas

8.1. Roles de pruebas

Los siguientes roles de prueba son necesarios en este proyecto:

1. Ingeniero de pruebas.
2. Gerente de pruebas.
3. Soporte metodológico.
4. Soporte técnico.
5. Experto en dominio.
6. Gerente de configuración de pruebas.

8.2. Personal de Pruebas

9. Entregables de pruebas

9.1. Documentación del proyecto

Se producirán los siguientes documentos durante el proyecto de prueba:

1. Plan de pruebas: el documento inicial y todas sus versiones anteriores/futuras.

Función	Nombre	Equivalente a Tiempo Completo
Gerente de Pruebas	T. Testware	0.20
Probador	P. Testcase	1.00
Probador	F. Testscript	1.00
Soporte	G. Allround	1.00
Experto en Dominio	A. Experto	0.60 (durante la ejecución de pruebas)

Cuadro 9: Funciones y equivalente a tiempo completo del personal de pruebas

2. Informes de defectos: se informan todos los defectos observados.

3. Informes semanales: informes de progreso realizados por el gerente de pruebas.

4. Recomendación para la liberación: formalmente es el final de la fase de ejecución de pruebas.

5. Informe de revisión: este informe da una evaluación del proceso de pruebas durante el proyecto.

9.2. Testware

Los siguientes documentos son entregables del proyecto de prueba:

1. Guion de pruebas: una descripción de cómo se realiza la prueba. Contiene acciones y verificaciones, relacionadas con casos de prueba, indicando la secuencia de ejecución.

2. Escenario de pruebas: un micro-plan de pruebas que coordina la ejecución de varios guiones de prueba y asigna probadores a los guiones de prueba.

3. Conjunto de datos inicial: archivos y conjuntos de datos necesarios para iniciar ciertas pruebas.

9.3. Almacenamiento

La estructura de directorios mostrada en la Tabla 10 está implementada en el servidor central de Solidly Embedded Ltd, el directorio se almacena en el directorio de gestión de productos: `\\PROD_MANAG`.

Directorio	Contenido
ACC_TEST_REALITA	Directorio principal del proyecto
PROCDOC	Documentación del proyecto
WORK_TESTWARE	Directorio de trabajo para testware
FINAL_TESTWARE	Directorio para archivar testware
DEFECTS	Base de datos para el almacenamiento de defectos
OTHER	Todos los demás documentos utilizados o producidos

Cuadro 10: Estructura de directorios

10. Gestión de configuración

10.1. Control del proceso de pruebas

El progreso de las pruebas y el agotamiento del presupuesto y el tiempo se monitorean y informan durante el proyecto de pruebas. Esto se realiza semanalmente y los resultados se informan en la reunión semanal de progreso.

10.2. Gestión de defectos

Para la gestión de defectos, se utiliza la herramienta "DefectTracker", se sigue el procedimiento estándar de defectos (SolEmb 76.1 "Procedimiento de defectos").

10.3. Métricas

El jefe de pruebas realiza un seguimiento de las siguientes métricas:

1. Número de defectos abiertos por categoría de gravedad en un momento dado.

2. Número de defectos resueltos en un período por categoría de gravedad.

3. Número total de defectos detectados.

4. Número de retrabajos por defecto.

5. Número total de retrabajos.

10.4. Ítems de gestión de configuración

El plan de pruebas está sujeto a la gestión de configuración, comenzando con la primera versión completa. El resto del testware se somete a la gestión de configuración después de la fase de finalización. Los cambios en la infraestructura de pruebas están sujetos a la gestión de configuración del departamento de soporte técnico de Solidly Embedded Ltd.

11. Bibliografía

1. Bart Broekman, Edwin Notenboom. *Testing Embedded Software*. Addison-Wesley Professional, 2003. ISBN: 978-0201795238