Programación (ARM-Cortex)

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

Proyecto PAPIME PE109416

Manual para construir un robot de enjambre.

Reconocimiento (Raspberry Pi)

Este manual forma parte de una serie de documentos diseñados para guiar a los participantes del Taller de Robótica de Enjambre: Nivel 3 – (4) PROGRAMACIÓN Y PLACA BASE. Este tercer taller práctico tiene como meta programar los microcontroladores ARM-Cortex, usando bibliotecas en lenguaje C y Sistema Operativo Linux.

Manual para construir un robot de enjambre.

Reconocimiento (Raspberry Pi)

Definición de Robot.

En su muy agradable libro introductorio sobre el tema de la robótica llamado *The Robotics Primer*, Maja J. Mataric [1] utiliza la siguiente definición: "*Un robot es un sistema autónomo que existe en el mundo físico, puede detectar su ambiente, y puede actuar para lograr algunos objetivos* ". La primera parte dice: "*Un robot es un sistema autónomo*". Por autónomo, queremos decir que un robot toma decisiones por sí mismo; no está controlado por un humano. y elimina cualquier máquina que esté controlada por alguien (a diferencia por ejemplo de una lavadora).

Los Robots de los que hablaremos a lo largo de este manual a veces pueden tener algún tipo de función remota, que permite a un ser humano controlarlo a distancia, pero esta funcionalidad generalmente está incorporada como una especie de medida de seguridad para diagnosticar un posible problema. Sin embargo, el objetivo principal es construir robots que puedan ser capaces de actuar y funcionar por su cuenta.

Fig. 3.1 Un robot puede tomar decisiones por sí mismo.

Continuando con la definición, necesitamos la otra parte de la oración que dice "*que existe en el mundo físico*", para diferenciarlos de los llamamos "robots de software" o simplemente "bots", por lo que un robot real necesita saber qué está sucediendo en el entorno

Introducción al diseño con Microcontroladores

Este proyecto tiene como objetivo hacer un manual para el diseño de robots de enjambre usando hardware informático de código abierto para microcontroladores de 32 bits con arquitectura ARM (**Advanced RISC Machines**)

En la actualidad, los robots de código abierto aún no son lo suficientemente sofisticados para llevar a cabo gran parte de las tareas de una persona, sin embargo, las estrategias de visión y la computación evolutiva es una oportunidad de desarrollo para los nuevos ingenieros que usan las herramientas tecnológicas para generar un perfil profesional que los hace valiosos en cualquier compañía.

en que opera. Además, esta es la razón por la cual la siguiente parte de la definición dice, "*puede sentir su entorno*".

Detectar lo que está sucediendo alrededor de un robot real es posiblemente la más importante característica. Para detectar sus entornos circundantes, los robots generalmente tienen sensores. Estos son dispositivos que miden las características físicas del ambiente y proporcionar esta información de vuelta al robot para que pueda, por ejemplo, reaccionar ante un ataque repentino cambios de temperatura, humedad o presión. Si miramos las semejanzas entre los robots y los humanos, podemos pensar en subsistemas sensores como reemplazos artificiales para órganos humanos que proporcionan información para el cerebro.

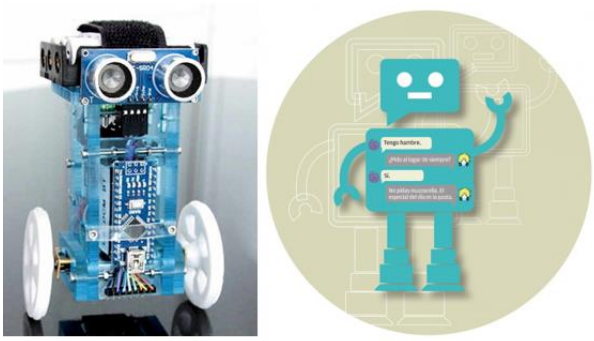


Fig. 3.2 Bots vs Robots.

Una consecuencia importante de esta definición es que cualquier cosa que no tenga sentido en su entorno no se puede llamar un robot. Esto incluye cualquier dispositivo que simplemente conduzca a ciegas o moverse de forma aleatoria, porque no utilizan la información del entorno en el que puede basar su comportamiento. Nuestra definición de robot refleja esto en su última parte cuando dice, "*puede actuar en consecuencia para lograr algunos objetivos* ".

Actuar sobre el medio ambiente puede sonar como una tarea muy compleja para un robot, pero en este caso, solo significa cambiar el mundo de alguna manera (incluso muy leve). Llamando estas partes del robot como efectores, y haciendo una similitud con el ser humano su equivalente artificial podrían ser las manos, piernas y otras partes del cuerpo que le permiten moverse. Los efectores hacen uso de algunos sistemas de nivel inferior como como motores o músculos que realmente llevan a cabo el movimiento.

Aunque los actuadores artificiales parecen funcionar de forma similar a los biológicos, una mirada más cercana revelará que en realidad son bastante diferentes, ya que no solo se trata de que el robot actúa sobre el medio ambiente, sino también sobre el logro de algunos objetivos. La mayoría de los robots se construyen para llevar a cabo (o, preferimos decir, para ayudar con) algunas tareas, como mover piezas pesadas en una fábrica o localizar víctimas en áreas afectadas por desastres naturales.

Sin embargo, el objetivo de un robot no necesariamente tiene que ser algo tan complejo y ambicioso como "realizar el trabajo duro para humanos". Puede ser algo simple, como" no toparse con obstáculos " o "enciende el interruptor de la luz".

Fig. 3.3 Robots que cumplen una tarea.

El uso de los microprocesadores y microcontroladores en la robótica.

En los módulos 1 y 2 de este manual se hace referencia al diseño y construcción de la parte física del robot. Y ahora analizaremos la parte de la percepción del robot, si no fuera por los controladores, un robot nunca podría ser completamente autónomo. De tal manera que se puedan usar datos de sensores para decidir qué hacer a continuación y luego ejecutar algunas acciones usando efectores. Esto puede parecer una descripción simple, pero al final, resulta que los controladores son bastante difíciles de programar, especialmente cuando se utilizan por primera vez, por lo que, se iniciará con los conceptos generales, la selección del controlador y el uso del ambiente de desarrollo para finalmente programarlo y realizar las tareas mencionadas.

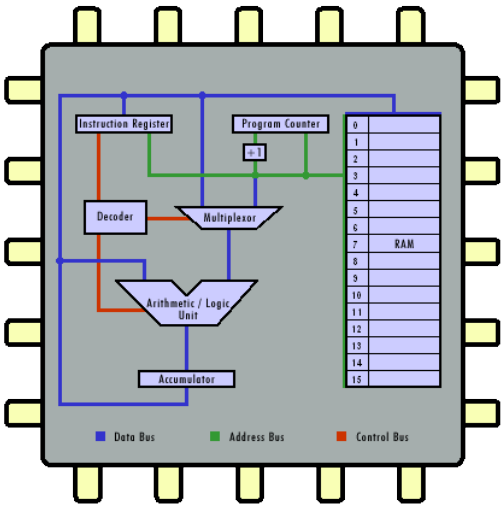


Fig. 3.4 Microprocesador.

Comenzaremos por definir a un microprocesador como un solo circuito integrado de muy alta escala de integración que en general se conoce como hardware. Y se suele llamar por analogía el «cerebro» de un ordenador. Tiene como características principales la programabilidad y la universalidad, por ser un dispositivo electrónico que se puede usar en muchas aplicaciones. De tal manera que al ejecutar una serie de instrucciones secuenciales que en general se conoce como software, puede ser reprogramado para

realizar tareas desde encendido y apagado de relevadores en un tiempo determinado hasta cálculos aritméticos de alta precisión.

El microprocesador se conecta a un circuito de reloj, normalmente basado en un cristal de cuarzo capaz de generar pulsos a un ritmo constante, de modo que genera varios ciclos (o pulsos) en un segundo. Este reloj, en la actualidad, genera miles de megahercios.

Desde el punto de vista lógico, singular y funcional, el microprocesador está compuesto básicamente por: varios registros, una unidad de control, una unidad aritmético-lógica, y dependiendo del fabricante, puede contener una unidad de coma flotante.

El microprocesador ejecuta instrucciones almacenadas como números binarios organizados secuencialmente en la memoria principal. Cuando al microprocesador se le conectan mas dispositivos de entrada y salida se dice que es una microcomputadora. Cuando el término se introdujo por primera vez a fines de la década de 1970, se refería a una computadora con un único microprocesador como su CPU, es decir, la computadora personal (PC).

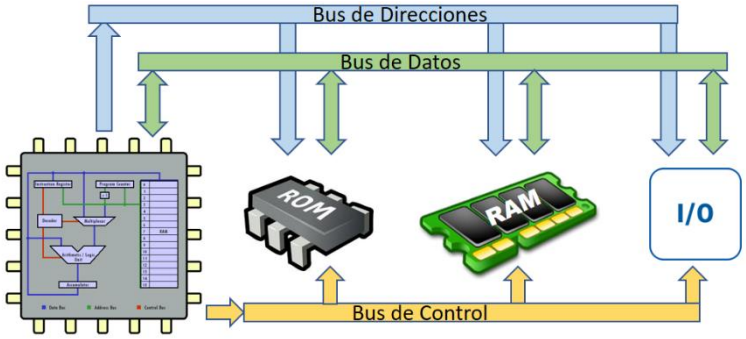


Fig. 3.5. Microcomputadora.

Se puede afirmar que un Microcontrolador es una microcomputadora en un único chip que contiene el procesador (la CPU), memoria no volátil para el programa (ROM o flash), memoria volátil para entrada y salida (RAM), un reloj y una unidad de control de E / S. Disponible en numerosos tamaños y

arquitecturas, miles de millones de unidades de microcontroladores (MCU) se incrustan cada año en productos desde juguetes hasta electrodomésticos y automóviles.

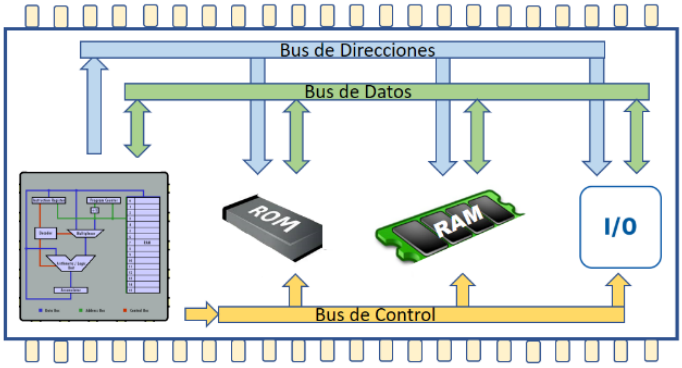


Fig. 3.6. Microcontrolador.

Ficha 3.1

Actividad 3.1.

Se sugiere revisar estos conceptos en referencias especializadas, así como los fundamentos e historia de los microprocesadores y microcontroladores.

Ver Material de clase. Tema I

Contraste entre la arquitectura Von Neumann y Harvard.

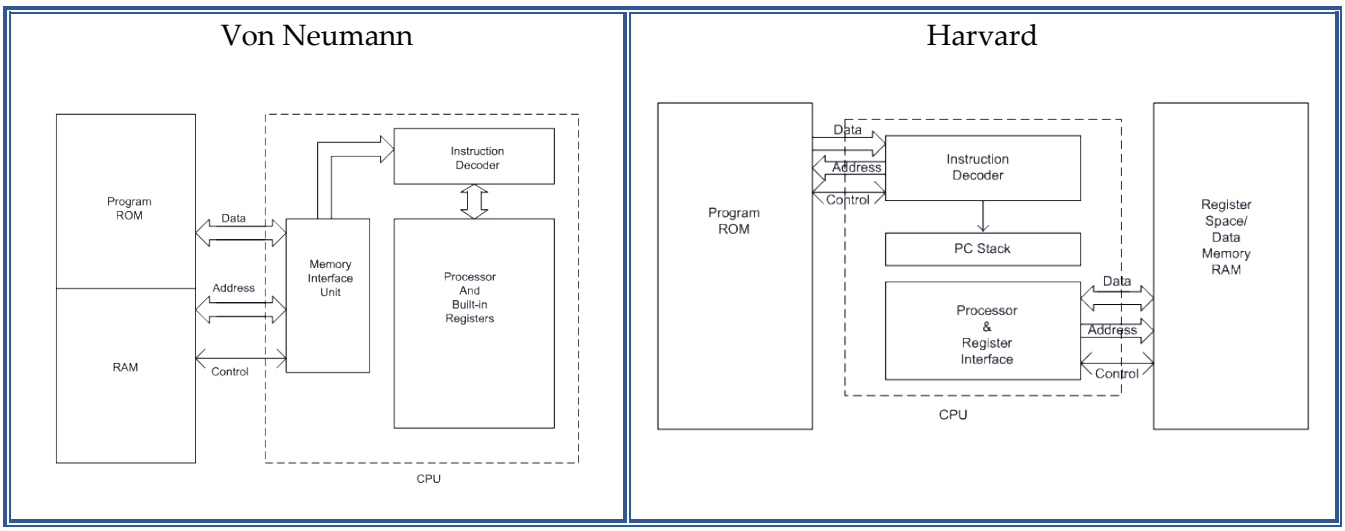
A fines de la década de 1940, el gobierno de los EE. UU solicitó a las universidades de Harvard y Princeton que crearan una arquitectura informática para utilizar en el cálculo de distancias de la artillería naval para aplicaciones de defensa. Princeton sugirió la arquitectura de la computadora con una única interfaz de memoria. También se conoce como la arquitectura Von Neumann después del nombre del científico jefe del proyecto.

Harvard sugirió una computadora con dos interfaces de memoria diferentes, una para los datos / variables y la otra para el programa / instrucciones. Aunque la arquitectura de Princeton fue aceptada por su simplicidad y facilidad de implementación, la arquitectura de Harvard se hizo popular más tarde, debido al paralelismo de la ejecución de instrucciones.

Aunque la tecnología de proceso y de encapsulado son vitales en la elaboración de procesadores más rápidos, es la arquitectura del procesador lo que hace la diferencia entre el rendimiento de una CPU y otra. Así que, dependiendo de cómo el procesador almacena los operandos en la memoria, existen tres

tipos de juegos de instrucciones: basadas en pilas, en el acumulador y basadas en registros. Las arquitecturas RISC (Conjunto Reducido de Instrucciones) y CISC (Conjunto Complejo de Instrucciones) son ejemplos de CPU con un conjunto de instrucciones para arquitecturas basadas en registros. Y debido al paralelismo, la arquitectura de Harvard ejecuta más instrucciones en un tiempo dado en comparación con la arquitectura de Von Neumann.

Tabla 3.1 Arquitectura Harvard vs. Von Neumann.



Los Microcontroladores en los sistemas de control.

Los microprocesadores se utilizan principalmente con fines computacionales, mientras que los microcontroladores encuentran una amplia aplicación en dispositivos que necesitan un procesamiento en tiempo real. Si tratamos de ver las partes de un robot de manera abstracta, esencialmente hay tres procesos importantes: detección (hecho por sensores), actuación (hecho por efectores), y planeación (hecho por controladores). Dependiendo de cómo juntemos estos tres procesos (ya que son los bloques de construcción también se les llama primitivos), podemos obtener diferentes arquitecturas con diferentes propiedades, también llamadas paradigmas.

Control reactivo

El control reactivo es probablemente la arquitectura (o paradigma) más simple, en este paradigma, como podamos ver en la siguiente figura, no hay un proceso de planificación involucrado. Hay una conexión directa entre la detección y la actuación, lo que significa que tan pronto como entran los datos del sensor, los efectores actúan sobre el medio ambiente de alguna manera predefinida:

Fig. 3.7. Control reactivo.

Así como los reflejos en nuestro cuerpo no envían la información al cerebro sobre algo que está pasando (que sería bastante lento), sino más bien simplemente a la médula espinal que es más cercana para que la respuesta sea rápida. Entonces el control reactivo del robot no tendrá ningún cálculo complejo, sino acciones rápidas y precalculadas que serán almacenadas.

Control jerárquico (deliberativo)
Supongamos que estás programando un robot de juego de ajedrez, seria genial si hiciera alguna planificación sobre el futuro para que pueda anticipar los turnos del oponente y luego ajustar su estrategia, basado en el turno actual del oponente. Una configuración como esta será perfecta para el paradigma de control jerárquico (o deliberativo). Como puede ver en la siguiente figura, el ciclo de planificación, actuación y detección es cerrado. Por lo tanto, el sistema puede moverse activamente hacia su objetivo, sea lo que sea:

Fig. 3.8. Control deliberativo.

Control híbrido
Los paradigmas de control anteriores son rápidos, pero no muy flexibles, o inteligente. Lo que realmente necesitaremos en muchos casos es algo intermedio. Y esto es precisamente lo que un paradigma de control híbrido intenta ofrecer. ¿Cómo podemos usar esto en un robot de la vida real? Supongamos que queremos construir un robot que tenga un cierto comportamiento (por ejemplo, las hormigas que recolecta y almacena comida). Tal robot definitivamente necesitaría tener su propia representación de su ubicación y donde debe llevar lo que recolecta

Una vez que se le haya dado la tarea de entregar el alimento, tendrá que planificar su camino y luego moverse junto a ese camino. Como podemos esperar, quizás haya otros elementos que también realizan la misma función. No podemos dejar que nuestro robot choque con cualquier objeto, y mucho menos dirigirse a otro destino Para esto, necesitamos un controlador reactivo bien ajustado, en la siguiente figura se muestra el esquema del paradigma de control híbrido y podemos ver que el robot al principio planifica su tarea, pero la divide en una serie de acciones que pueden ser ejecutadas por el paradigma reactivo. Una cosa interesante de notar aquí es el hecho de que los datos sensoriales están disponibles para ayudar a la planificación y las partes que actúan del sistema:

Fig. 3.9. Control híbrido.

La aplicación de microcontroladores es numerosa, desde aplicaciones domésticas, como lavadoras, televisores, aire acondicionado, hasta su utilización en automóviles, también en la industria de control de procesos, teléfonos celulares, unidades eléctricas, robótica y en aplicaciones espaciales.

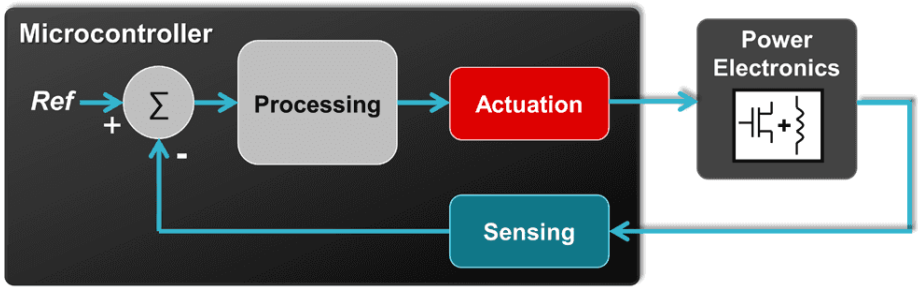


Fig. 3.10. El microcontrolador en los sistemas de control.

Los fabricantes de semiconductores ofrecen una gran variedad de microcontroladores de 8, 16 o 32 bits, según la aplicación que se desee controlar.

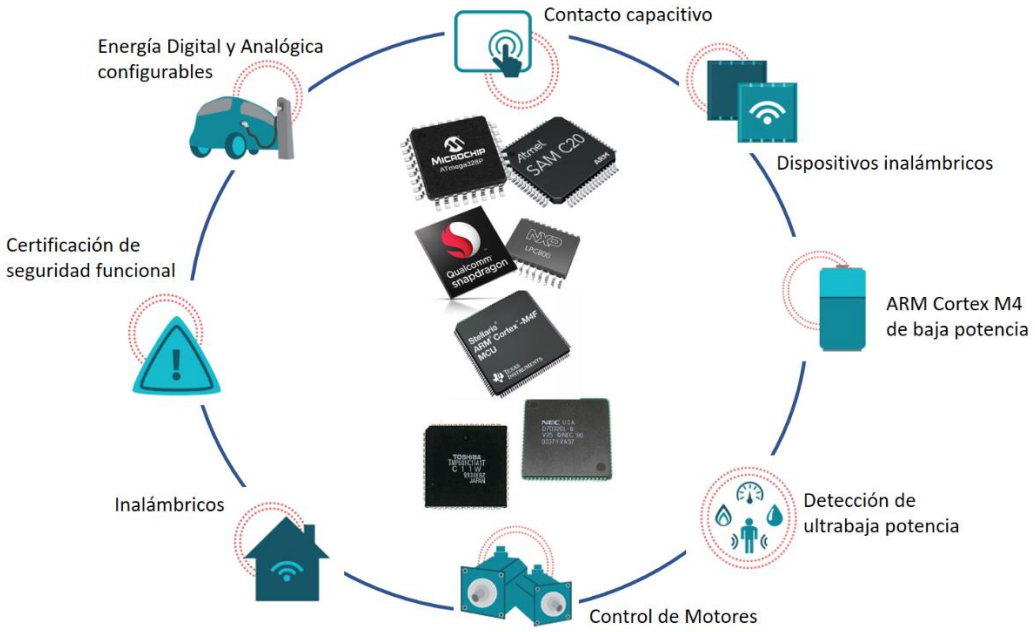


Fig. 3.9. Fabricantes y familias de circuitos para las principales aplicaciones en la industria.

Ficha 3.2

Actividad 3.2.

Se sugiere visitar los sitios de los fabricantes más populares y revisar modelos y costos relacionados con los microcontroladores de 32 bits con arquitectura ARM (**Advanced RISC Machines**) Series Cortex-M.

Ver Wikipedia ARM Cortex-M

Tarjeta de desarrollo Tiva-C LaunchPad.

Las tarjetas de desarrollo son típicamente placas de E / S, que pueden enviar señales de control en forma de pulsos digitales a la placa controladora de motores y puede recibir entradas de varios tipos de sensores También se pueden interconectar los codificadores de motores a la placa de control para la retroalimentación del movimiento.

Los LaunchPads Tiva-C (también conocidos como TM4C) [2] son microcontroladores autónomos de bajo costo, que cuentan con una CPU ARM Cortex-M4F de 32 bits que opera de 80 a 120 MHz, fabricados por Texas Instruments. Las E/S digitales y analógicas pueden configurarse fácilmente, permitiendo una gran variedad de aplicaciones, también los múltiples puertos seriales ofrecen la capacidad de interactuar con más elementos como tarjetas de prueba u otros módulos de comunicaciones.

El reloj de 80 o 120 MHz, hace que sea de 5 a 7 veces más rápida que el microcontrolador ATmega328P de 16 MHz del Arduino Uno. Al igual que con cualquier Cortex M4, la CPU tiene algunas instrucciones DSP (procesador de señal digital). Y la CPU contiene una unidad de coma flotante, así como un puerto USB adicional lo que permite la conexión de múltiples dispositivos, también tienen LED (s) RGB, que le permite generar varios colores combinando los tres colores básicos (rojo, azul y verde).

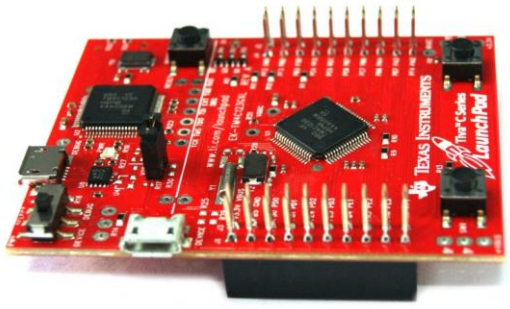


Fig. 3.10. Tiva-C Tm4c123gh6pm.

Ficha 3.3

Actividad 3.3.

Se sugiere leer el instructivo de la caja y correr el demo para revisar su correcto funcionamiento.

Ver **ReadMe First (Rev. A)**

Programación (Lenguaje C)

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

Proyecto PAPIME PE109416

Programación (Lenguaje C)

El uso de lenguaje C para programar Microcontroladores

El lenguaje C dispone de todas las ventajas de un lenguaje de programación de alto nivel y le permite realizar algunas operaciones tanto sobre los bytes como sobre los bits (operaciones lógicas, desplazamiento etc.). Las características de C pueden ser muy útiles al programar los microcontroladores. Además, C está estandarizado (el estándar ANSI), es muy portable, así que el mismo código se puede utilizar muchas veces en diferentes proyectos. Lo que lo hace accesible para cualquiera que conozca este lenguaje sin reparar en el propósito de uso del microcontrolador. C es un lenguaje compilado, lo que significa que los archivos fuentes que contienen el código C se traducen a lenguaje máquina por el compilador.

¿Por qué programar microcontroladores en 'C'?

- Código más compacto (visualmente hablando)
- Código menos críptico: más fácil de entender
- Más fácil de mantener / actualizar
- Más fácil de administrar grandes proyectos con múltiples programadores
- Más portátil (hasta cierto punto)

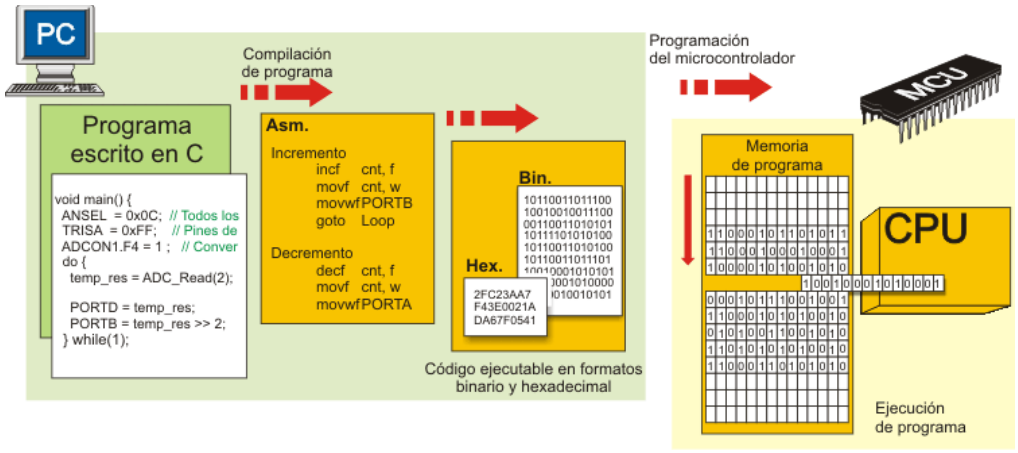


Fig. 3.11. Proceso de compilación de un programa en C.

Ficha 3.4

Actividad 3.4.

Introducción al lenguaje C (diapositivas)
Se sugiere ver el entrenamiento en línea de Ansi C Parte 1, de Freescale Cup México.

TivaWare

El software TivaWare para la Serie C es un conjunto extenso de herramientas de software diseñadas para simplificar y acelerar el desarrollo de las aplicaciones de MCU basadas en la Serie Tiva C. Todo el software TivaWare para la Serie C tiene una licencia gratuita y permite el uso libre de regalías para que los usuarios puedan crear aplicaciones.
Se recomienda descargar e instalar los laboratorios del sitio wiki. El material se instalará en: *C:\Tiva_TM4C123G_LaunchPad* y el manual del taller para aprender a usar las bibliotecas de TivaWare se puede descargar de: *http://software-dl.ti.com/trainingTTO/trainingTTO_public_sw/GSW-TM4C123G-LaunchPad/TM4C123G_LaunchPad_Workshop_Workbook.pdf*

Es importante realizar la sección "***Add Path and Build Variables***" del **Lab 2** para usar las bibliotecas de TivaWare en los siguientes programas.

Ficha 3.5

Actividad 3.5.

Se sugiere realizar los Laboratorios:
Interrupts and the Timers
ADC12
USB
UART
PWM

SISTEMA DE EQUILIBRIO DE UN ROBOT MOVIL

Una unidad de medición inercial (IMU) es un dispositivo electrónico que mide velocidad, orientación y fuerzas gravitacionales usando una combinación de acelerómetros, giroscopios y magnetómetros. Un IMU tiene muchas aplicaciones en robótica; algunos de las aplicaciones están en equilibrio de vehículos aéreos no tripulados (UAV) y navegación de robots.

Para este segundo diseño, se requiere implementar un IMU para la navegación de un robot móvil de tipo diferencial. El robot que se requiere diseñar es un robot con ruedas diferenciales, donde el movimiento se basa en dos ruedas accionadas por separado colocadas en ambos lados del cuerpo del robot. Puede cambiar su dirección cambiando la velocidad relativa de rotación de sus ruedas, y por lo tanto, no requiere un movimiento de dirección adicional. Para equilibrar el robot, se puede agregar una rueda giratoria libre o ruedas giratorias.

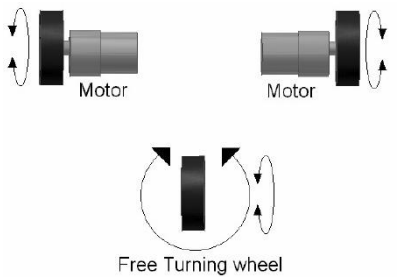


Fig. 3.12. Robot diferencial.

Navegación inercial

Una IMU proporciona aceleración y orientación relativa al espacio inercial, si conoce la posición inicial, la velocidad y la orientación, puede calcular la velocidad integrando la aceleración detectada y la segunda integración da la posición. Para obtener la dirección correcta del robot, se requiere la orientación del robot; esto se obtendrá integrando la velocidad angular detectada del giroscopio.

La siguiente figura ilustra un sistema de navegación inercial, que convertirá los valores IMU a datos odométricos:

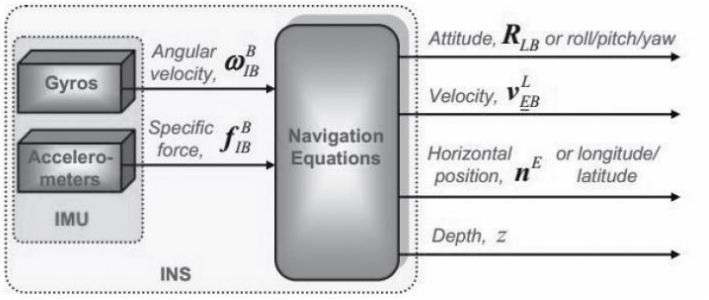


Fig. 3.13. Sistema de navegación inercial.

Los valores que obtenemos del IMU se convierten en información de navegación usando ecuaciones de navegación y alimentación en filtros de estimación como el filtro de Kalman. Este tipo de filtro usa un algoritmo que estima el estado de un sistema del datos medidos (http://en.wikipedia.org/wiki/Kalman_filter). Los datos del sistema de navegación inercial (INS) tendrá cierta deriva debido al error del acelerómetro y el giroscopio. Para limitar la deriva generalmente es necesario otros sensores que proporcionan mediciones directas de las cantidades integradas. Basado en mediciones y modelos de error del sensor, el filtro de Kalman estima errores en las ecuaciones de navegación. La siguiente figura muestra un diagrama de un sistema de navegación inercial asistida que utiliza el filtro de Kalman:

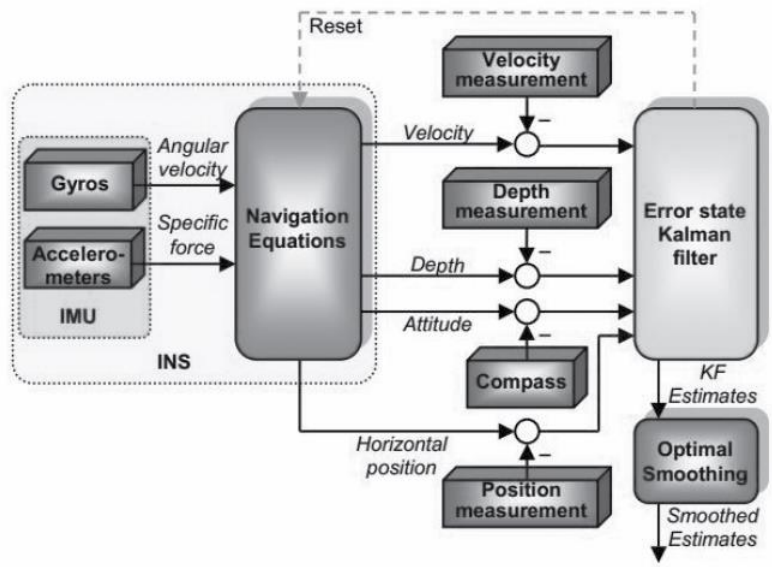


Fig. 3.14. Sistema de navegación inercial y filtro Kalman.

Junto con los codificadores del motor, el valor de la IMU se puede tomar como el valor del odómetro y se puede utilizar para estimar la posición actual de un objeto en movimiento usando una posición previamente determinada.

Interfaz entre MPU 6050 y el microprocesador

La familia de piezas MPU-6000 / MPU-6050 están diseñados para un bajo consumo de potencia, bajo costo y alto rendimiento. Actualmente se usan en teléfonos inteligentes, tabletas, sensores portátiles y robótica. Los dispositivos MPU-6000/6050 combinan un giroscopio de 3 ejes y un acelerómetro de 3 ejes.

Fig. 3.15. MPU-6050.

El MPU 6050 se puede conseguir en el siguiente enlace:

https://www.sparkfun.com/products/11028

Fase de análisis

Seguridad. Ningún riego para el humano, una vez inhabilitado el robot móvil se debe deshacer como desechos electrónicos.
Precisión. El sistema de posicionamiento y la unidad de medición inercial debe ser de precisión media.
Exactitud. Las mediciones deben ser confiables.
Resolución. La resolución debe ser adecuada para un sistema de edometría experimental.
Tiempo de respuesta. En tiempo real.
Ancho de Banda. Entre más rápido mejor, con limite en la transmisión de datos.
Mantenibilidad. Se deben usar dispositivos genéricos y de preferencia en lenguaje de alto nivel.
Testabilidad. Las pruebas deben considerar el despliegue de los datos en terminal.
Compatibilidad. Se debe ajustar a los estándares existentes.
Tiempo medio entre fallas. Debe diseñarse para una vida útil de por lo menos 5 años.
Tamaño y Peso. 20*20 cm^2 max. Y 1.5 kg max.
Energía. 12 v max, y 3.3 para las I/O digitales.
Costo de ingeniería no recurrente. $ 1,250.00 max.
Costo unitario. $ 300.00 aprox.
Tiempo del prototipo. 60 días max.
Tiempo de comercialización. No aplica.
Factores Humanos. 30 aprox.

Documento de Requisitos.

Información general.

Proyecto de robótica de enjambre, FES Aragón, UNAM. 2018

Objetivos.

Diseñar y construir un sistema de posicionamiento a través de una unidad de medición inercial (IMU), para un robot móvil inteligente para recolectar información de su entorno.

Proceso.

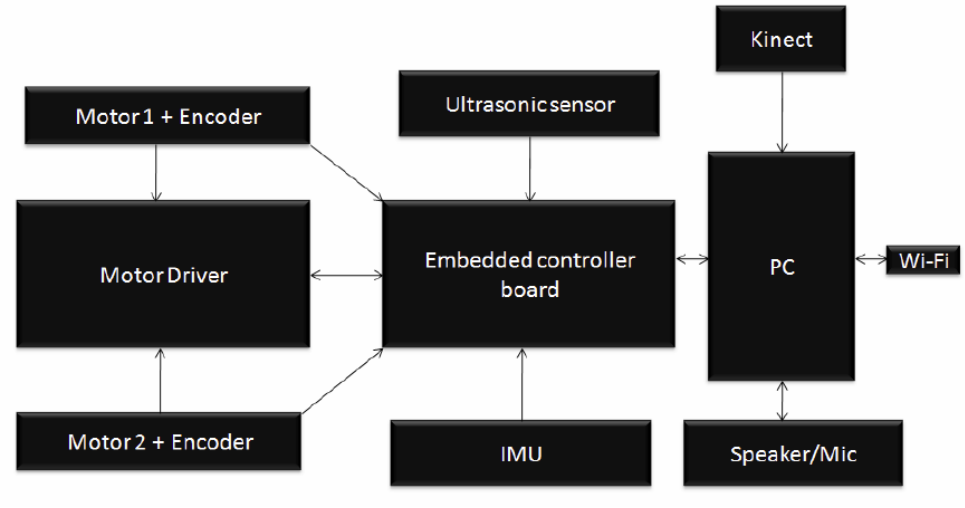


Fig. 3.16. Sistema de navegación inercial y filtro Kalman.

Refinamiento sucesivo.

¿Qué significa estar en un estado? Enumerar los parámetros del estado
¿Cuál es el estado inicial del sistema? Definir el estado inicial
¿Qué información necesitamos recolectar? Enumerar los datos de entrada
¿Qué información necesitamos generar? Enumerar los datos de salida
¿Cómo pasamos de un estado a otro? Especificar acciones que podríamos realizar
¿Cuál es el estado final deseado? Definir el objetivo final

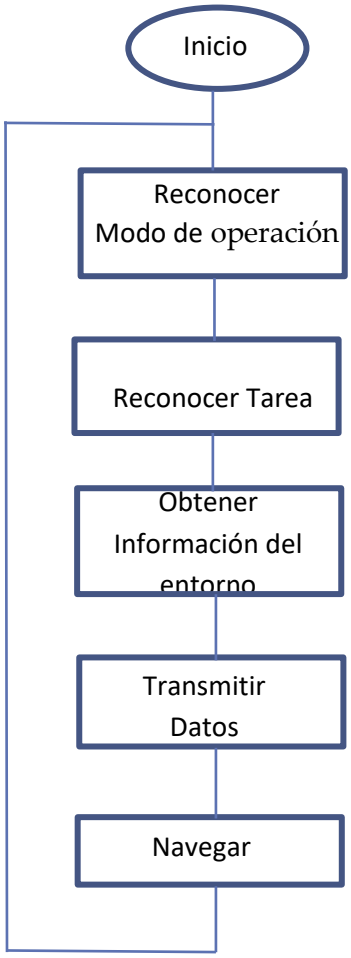


Fig. 3.17. Diagrama de flujo general.

Tipos de Sistemas Embebidos

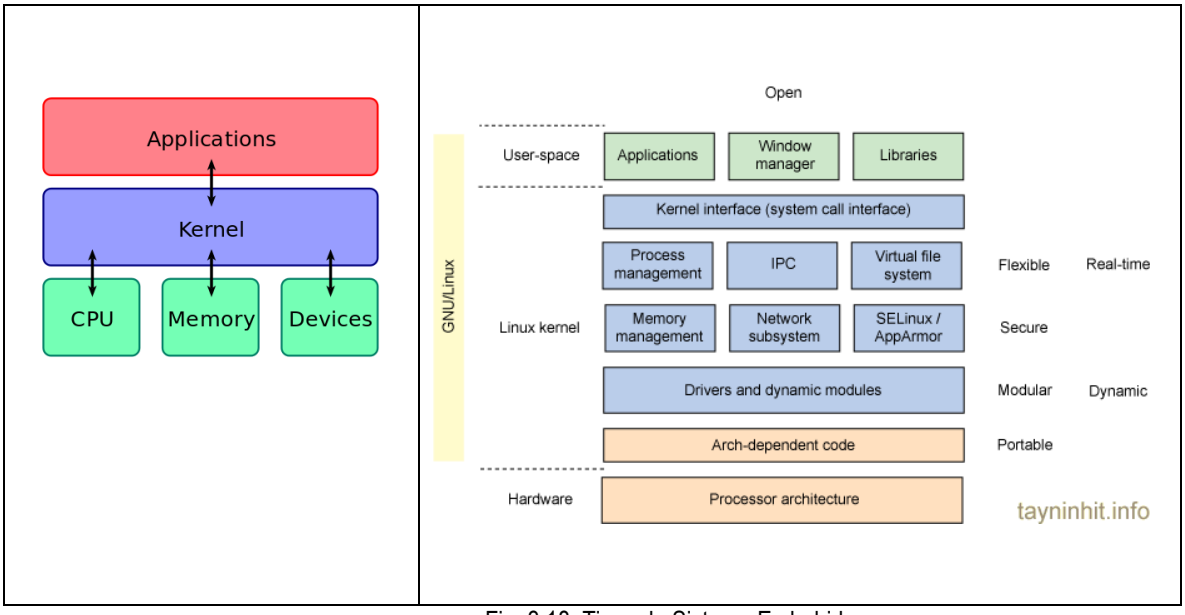


Fig. 3.18. Tipos de Sistema Embebidos.

Para este diseño se propone utilizar ambos tipos de sistemas embebidos. La primera arquitectura solo tiene como objetivo hacer el control del sistema de navegación del robot y posteriormente será parte de una aplicación en un sistema de alto rendimiento como Raspberry pi con raspbian o un dispositivo móvil con Android.

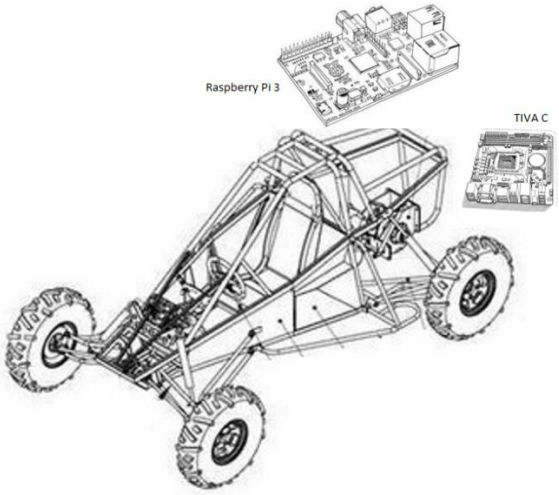


Fig. 3.19. Diseño del robot móvil más tarjeta de alto rendimiento.

SISTEMA DE DETECCIÓN DE OBSTACULOS

Una navegación ideal significa que un robot puede planificar su trayectoria desde su posición actual hasta el destino y puede moverse sin ningún obstáculo. Estamos usando sensores ultrasónicos para encontrar la distancia a un objeto, y para obtener los datos de odometría del robot; también usamos sensores de proximidad IR (rayos infrarrojos) para detectar los obstáculos y evitar colisiones.

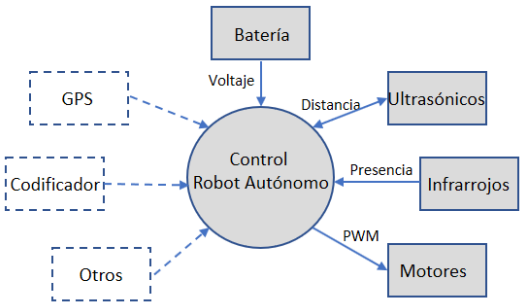


Fig. 3.20. Flujo de datos para el sistema de control.

El primer paso del desarrollo de firmware es el diseño de la interacción entre componentes, que se puede observar en el diagrama de contexto de la figura anterior, usando este diagrama se pueden observar las conexiones entre el programa de control y los otros componentes, así como la dirección del flujo de datos. Como se puede ver, la aplicación tiene que procesar el trabajo de muchos sensores, algunos necesitan comunicación bidireccional para hacer la inicialización, medición y lectura de datos; y en otros, sólo se trata de leer los valores. La siguiente tarea del control es activar los actuadores y la comunicación con un sistema superior en caso de ser requerida.

El siguiente diagrama, es considerado como el primer nivel de flujo de datos en la aplicación de control, y proporciona una mirada más cercana al objetivo específico de este trabajo. Describe la división básica en las tres partes (procesos: Sensores, Actuadores y Comunicación con el servidor) y sus conexiones. Estos procesos estarán trabajando independientemente durante la mayor parte del tiempo. Por lo tanto, tiene que haber cuellos de botella entre ellos. Usando la programación jerárquica la información de los sensores se guarda en un registro de memoria (Registro para sensores), que será compartida, y el proceso de comunicación con el servidor podrá leer estos datos. De igual forma se guardó en un registro de memoria para los codificadores y actuadores, logrando de esta forma triangular la información para un control más eficaz.

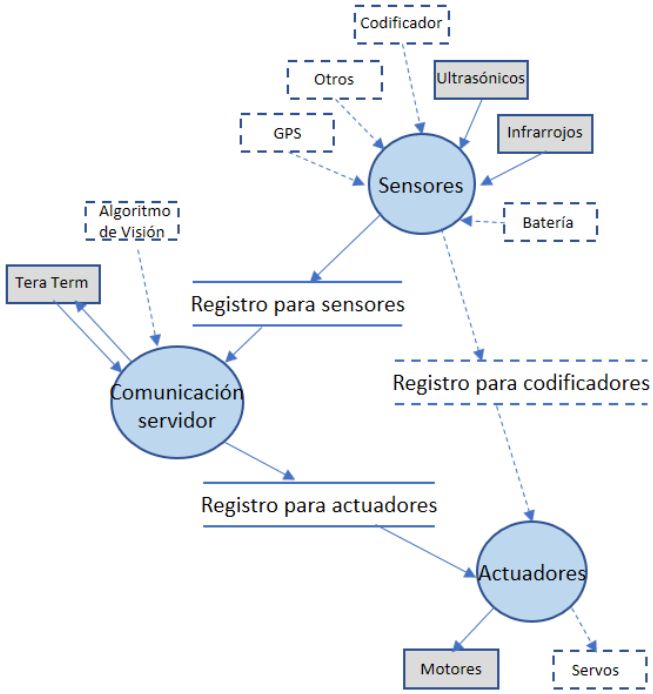


Fig. 3.21. Diagrama de flujo del sistema de control.

Primera Etapa – Comunicación.

El sistema de comunicación Bluetooth es un protocolo de comunicación que se ha desarrollado para la comunicación inalámbrica de dispositivos de bajo consumo. EL modulo Bluetooth **HC-05** viene configurado de fábrica como Esclavo, pero se puede cambiar para que trabaje como maestro, además al igual que el hc-06, se puede cambiar el nombre, código de vinculación velocidad y otros parámetros más.

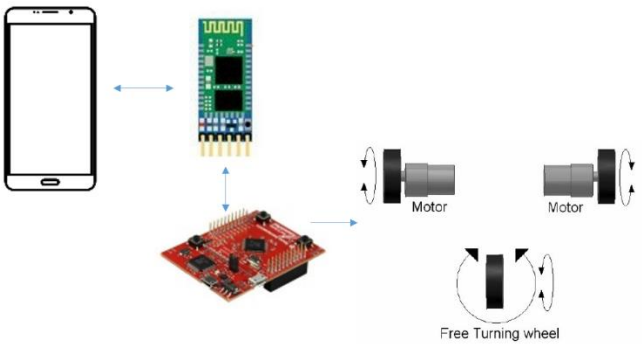


Fig. 3.22. Control usando Bluetooh

Aunque existen muchas Apps que conectan el módulo **HC-05**, se sugiere utilizar AppInventor para hacer un diseño que pueda ser modificado, o en el mejor de los casos programar en Java la aplicación con Android Studio.

Una vez terminada esta la etapa de comunicación el sistema está listo para ***Reconocer el Modo de operación***:

0 – Manual
1 – Automático

Segunda Etapa – Sensores.

El robot móvil deberá contar con varios sensores para determinar la presencia de un obstáculo y la distancia a la que se encuentra. Así como los valores x, y, z de la orientación relativa del IMU para su navegación inercial.

Se puede tomar como referencia el capítulo 6 del libro: Learning Robotics Using Python, Letin Joseph.

Para avanzar en el diseño es importante identificar que estados pueden hacerse en paralelo para optimizar los tiempos de entrega del proyecto. Por lo tanto, el diagrama de flujo del diseño inicial puede modificarse, con base en la técnica de refinamiento sucesivo, quedando de la siguiente forma:

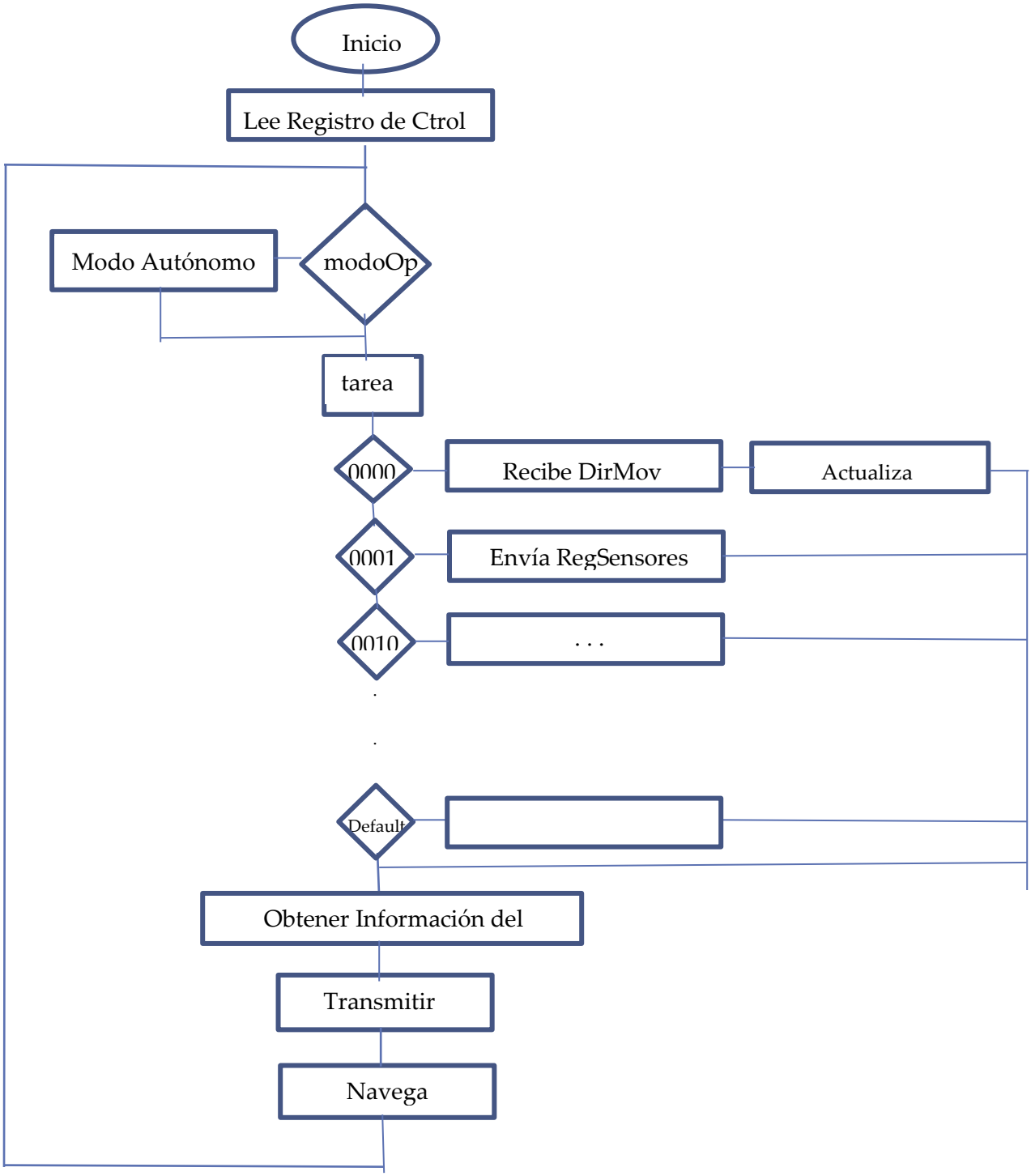


Fig. 3.23. Nuevo diagrama general. Refinamiento sucesivo.

Tareas del Sistema.

Registro de Control

-	-	-	Tbit3	TBit2	TBit1	TBit0	ModoOp

Bit 0 ModoOp 0 Manual
1 Automático

Bit 1 TBit0
Bit 2 TBit1 Tarea
Bit 3 TBit2
Bit 4 TBit3

Bit 5 -7 Sin usar

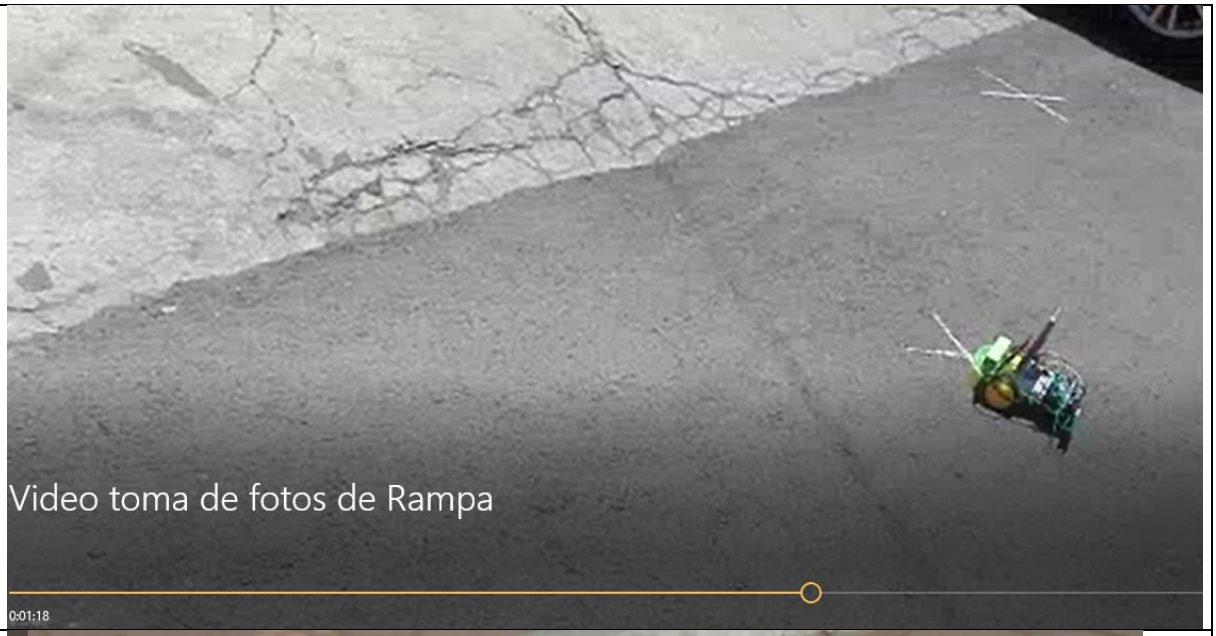
Ficha 3.6

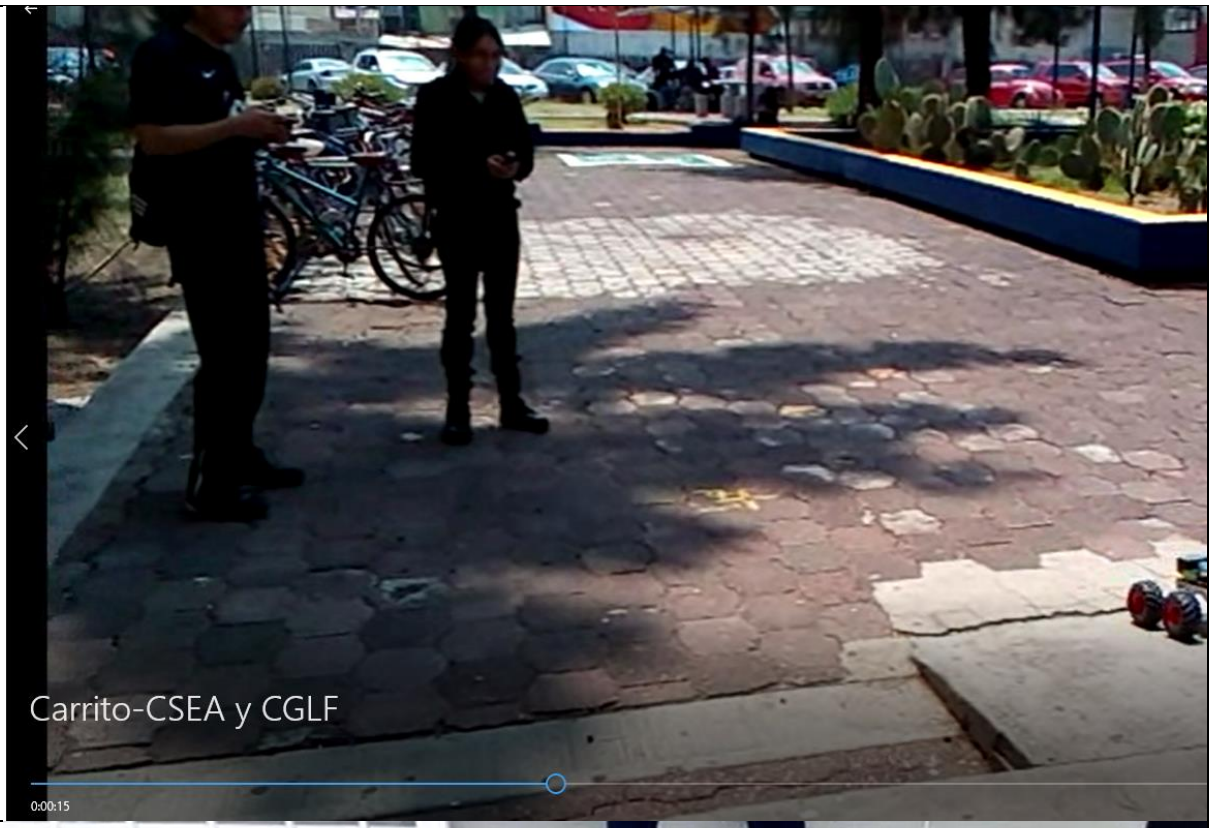
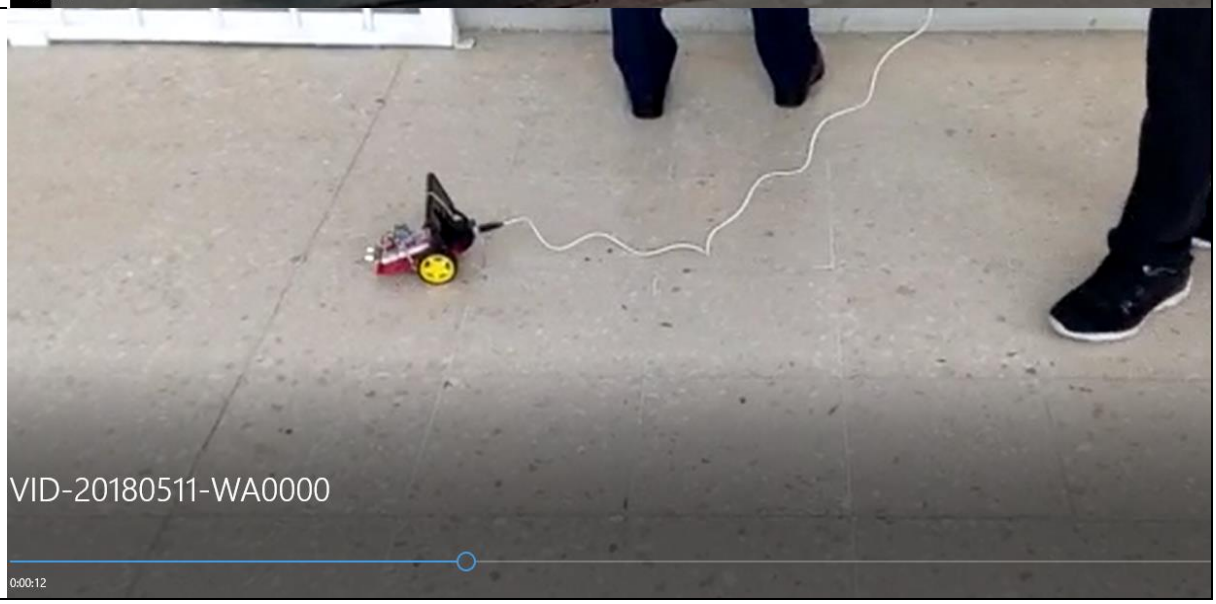
Actividad 3.6.

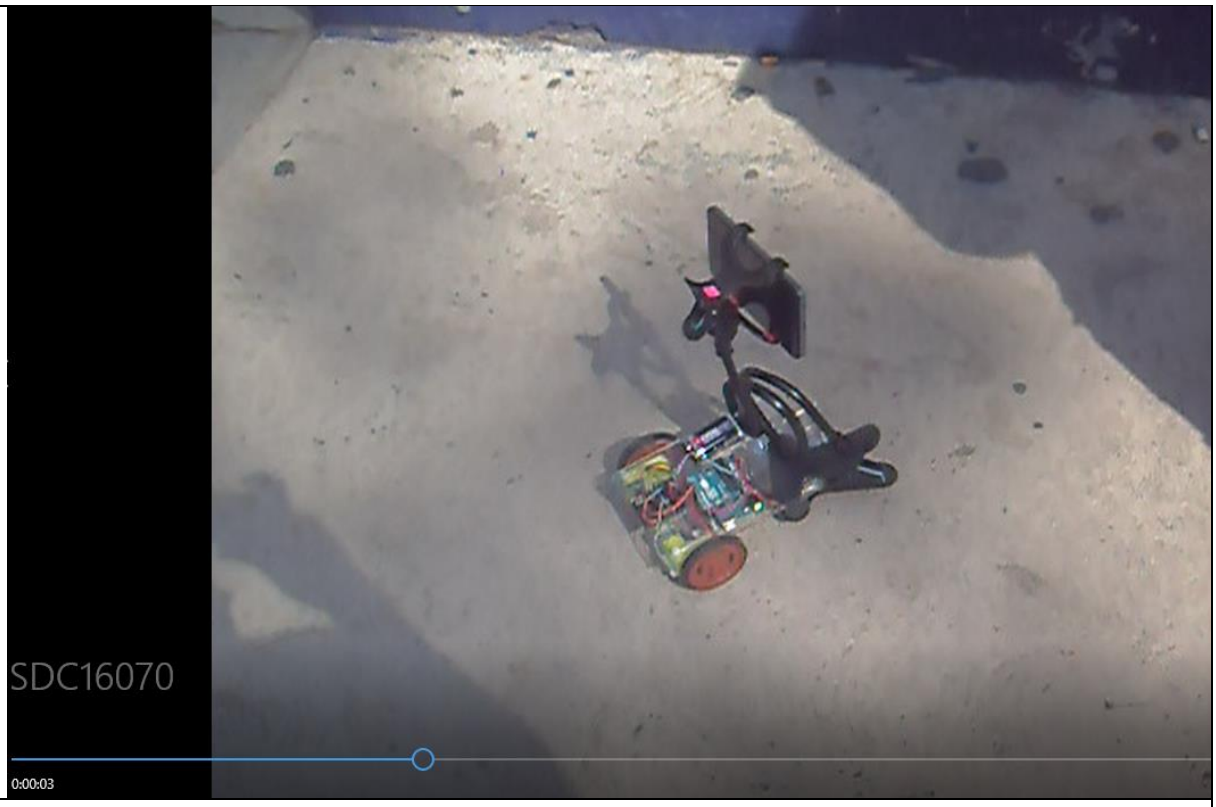
En cuanto el Robot Móvil se coloque en la posición exacta, se tomará una foto a través de un Smartphone para configurar una base de datos.

1. La posición del robot deberá estar aproximadamente de 1.0 m a 1.5 m de la rampa.
2. La altura del suelo al lente de la cámara del Smartphone deberá ser de 25 cm.
3. El formato de la imagen deberá ser jpg.
4. La resolución del tamaño de la imagen es libre.
5. Se deberán tomar 3 fotos en tres posiciones diferentes enfocando la rampa y 1 foto del entorno. Y repetir 3 veces con diferentes intensidades de luz.

Contribución Enjambre3.1

Robot móvil 1 tomando foto de rampa	
Robot móvil 2 tomando foto de rampa	

Robot móvil 3 tomando foto de rampa	
Robot móvil 4 tomando foto de rampa	

Robot móvil 5 tomando foto de rampa	

Reconocimiento (Raspberry Pi)

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

Proyecto PAPIME PE109416

Reconocimiento (Raspberry Pi)

Raspberry Pi.

La computadora de placa reducida **Raspberry Pi** es de bajo costo y fue desarrollada en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. La Raspberry Pi 3 contiene una CPU ARM Cortex-A53 de 1,2 GHz y, por lo tanto, es la primera versión compatible con la arquitectura arm64, lo que la convierte en una placa muy poderosa donde se puede ejecutar el sistema operativo Raspbian con todas las bondades de Linux.

Visión de Máquina.

La visión artificial se puede implementar en una máquina con capacidades de cámara y una comprensión de lo que son los objetos. La máquina usa su cámara para detectar objetos físicos en su entorno. La visión artificial o la visión por computadora tiene como objetivo adquirir, analiza y comprende una imagen fija o video. Este campo involucra conocimientos tales como procesamiento de imágenes, reconocimiento de patrones y aprendizaje de máquina.

Los avances en el campo de reconocimiento de patrones y aprendizaje de máquina nos ayudan a implementar sistemas electrónicos que perciban ciertos patrones en las imágenes, por ejemplo, personas o rostros. Desde una vista de reconocimiento de patrones y aprendizaje de máquina, debemos registrar a la persona para que la máquina pueda identificarla en una imagen existente. Para construir un sistema de visión artificial, utilizamos el diseño general que se muestra en la siguiente figura:

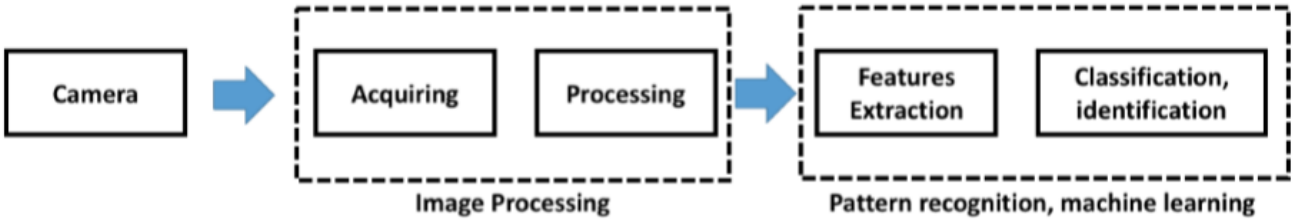


Fig. 3.24. Sistema de visión.

Ficha 3.6

Actividad 3.6.

Instalar OpenCV en Raspberry Pi:

a) Instalar el sistema operativo en la memoria SD.

b) Conectar a Raspberry.
Nota. Se recomienda activar Secure Shell (SSH) y activar el Wifi, para trabajar de manera remota.

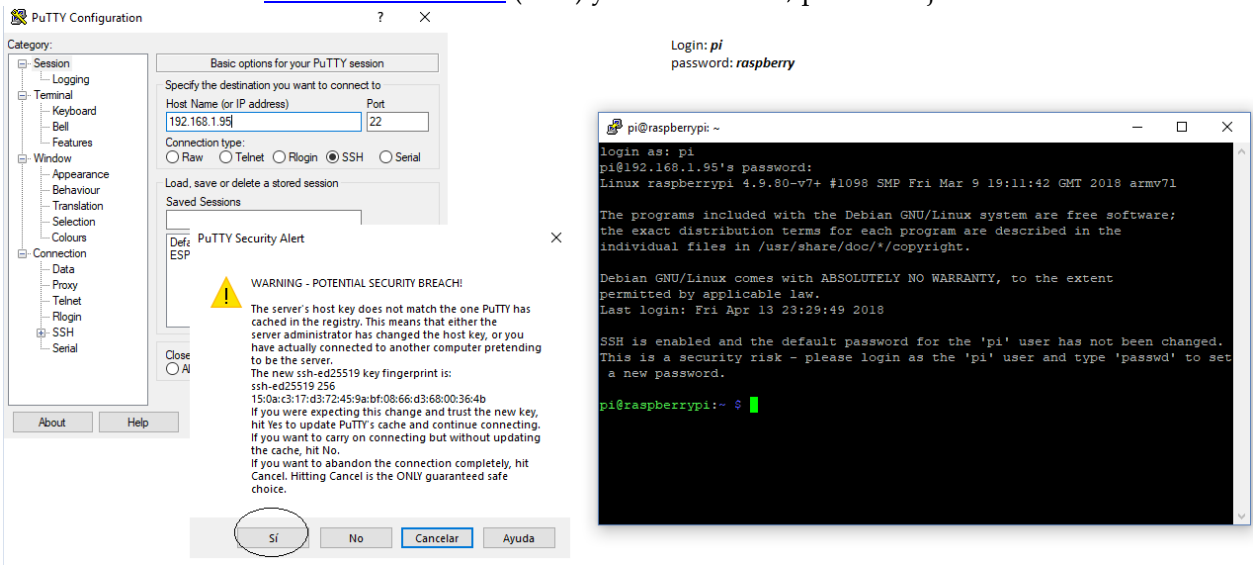


Fig. 3.25. Conexión grafica con Putty.

c) Desde una ventana de comando en Raspbian ejecutar los siguientes comandos:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential git cmake pkg-config libgtk2.0-dev
```

Ver Material de clase "Vision de Máquina con Raspberry Pi". Tema 3. Problema objetivo.

Procesamiento (OpenCV-Python)

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

Proyecto PAPIME PE109416

Reconocimiento (Raspberry Pi)

Sistemas Embebidos.

Se trata de un sistema de computación diseñado para realizar una o algunas funciones dedicadas frecuentemente en un sistema de computación en tiempo real. Al contrario de lo que ocurre con los ordenadores de propósito general (como por ejemplo una computadora personal o PC) que están diseñados para cubrir un amplio rango de necesidades, los sistemas embebidos se diseñan para cubrir necesidades específicas. En un sistema embebido la mayoría de los componentes se encuentran incluidos en la placa base (la tarjeta de vídeo, audio, módem, etc.) y muchas veces los dispositivos resultantes no tienen el aspecto de lo que se suele asociar a una computadora.

Los sistemas embebidos se pueden programar directamente en el lenguaje ensamblador del microcontrolador o microprocesador incorporado sobre el mismo, o también, utilizando los compiladores específicos, pueden utilizarse lenguajes como C/C++ o Python; en algunos casos, cuando el tiempo de respuesta de la aplicación no es un factor crítico, también pueden usarse lenguajes como JAVA.

Con los modernos sistemas PC embebida basados en microprocesadores i486 o i586 se llega a integrar el mundo del PC compatible con las aplicaciones industriales. Ello implica numerosas ventajas:

- Posibilidad de utilización de sistemas operativos potentes que ya realizan numerosas tareas: comunicaciones por redes de datos, soporte gráfico, concurrencia con lanzamiento de hilos, etc. Estos sistemas operativos pueden ser los mismos que para PC compatibles (Linux, Windows, MS-DOS) con fuertes exigencias en hardware o bien ser una versión reducida de los mismos con características orientadas a los PC embebidos.
- Al utilizar dichos sistemas operativos se pueden encontrar fácilmente herramientas de desarrollo software potentes, así como numerosos programadores que las dominan, dada la extensión mundial de las aplicaciones para PC compatibles.
- Reducción en el precio de los componentes hardware y software debido a la gran cantidad de PC en el mundo.

Ficha 3.7

Actividad 3.7.

a) Ejecutar los comandos básicos del sistema operativo Linux para poder navegar en el servidor: 132.248.173.84, usando Xming y PuTTy.
b) Realizar los ejemplos de OpenCV para la manipulación de imágenes.

UN ENFOQUE DE COMPETENCIAS.

La FES Aragón actualmente está pasando un proceso de Certificación en su Modelo de Enseñanza en sus carreras de Ingeniería, por lo que a continuación se propone una rubrica para calificar el nivel de comprensión de los integrantes del Taller:

Rúbrica de Evaluación – Los alumnos de Ingeniería de la FES Aragón Aplican, analizan y sintetizan procesos de diseño de ingeniería que resulten en proyectos que cumplen las necesidades especificadas.

Criterios de Desempeño	**Indicadores dimensionales**	**Tipo de Rúbrica:** Analítica.			
		Lo supera (10)	**Lo logra (8)**	**Parcialmente (6)**	**No lo logra (0)**
Realizar experimentos siguiendo el protocolo establecido.	Muestra los métodos y equipos seleccionados para la experimentación.	Explica métodos y equipos para la experimentación enfocada a los sistemas con microprocesador.	Muestra el experimento funcionando, pero no sabe cómo funciona el sistema con microprocesador.	Realiza experimentos siguiendo parcialmente el protocolo establecido.	No es capaz de seguir el protocolo establecido para la realización de experimentos.
Utiliza información experimental para el análisis, evaluación y diseño con Raspberry Pi.	Registra la eficiencia del diseño a través de simulaciones y pruebas en hardware y software.	Analiza información experimental relevante para realizar un nuevo diseño.	Utiliza información experimental para realizar el mismo proceso de diseño.	Clasifica la información experimental sin buenos resultados en las pruebas.	Ignora la información experimental en el diseño.

Experimento 3.1.

El laboratorio L3 necesita un sistema de seguridad en el área de servidores. Según la teoría de Machine Learning (Aprendizaje Automático) es necesario hacer una Recolección de Datos. La FES Aragón acude a ustedes Ingenieros en Electrónica para implementar un sistema de adquisición de imágenes, para hacer una clasificación Intruso/No Intruso

Construcción del Modelo.

Un equipo de estudiantes selecciona y modifica el programa para su implementación.

Documentar el modelo y proceso.

¿Cuáles son las herramientas que selecciono y por qué?

¿Qué herramienta matemática utilizo?

¿Qué características se deben considerar para hacer una buena recolección de imágenes?

¿Qué fuentes y recursos utilizo?

Refinación mediante autoevaluación

Notifique las problemáticas encontradas y su solución

Especifique las consideraciones de seguridad, costo y técnicas requeridas.

Generación del Modelo.

Se presentan resultados y posibles mejoras.

Efectividad

Análisis de costos y portabilidad.

Conclusiones técnicas, éticas y oportunidades.

Relación entre OpenCV y Hardware.

En el desarrollo de la visión por computador o aplicaciones de procesamiento de imágenes es común si no se requiere mucho procesamiento, se puede utilizar un ordenador portátil o un PC con GPU integrada, o si necesitamos una gran cantidad de poder de procesamiento se utilizan estaciones de trabajo con arquitectura NVIDIA, o incluso otros equipos con tecnologías especiales para procesamiento masivo de datos. Pero para sistemas embebidos, hay muchas más opciones a elegir en función de los requerimientos para solucionar una aplicación especifica. En la tabla 3.1 se pueden observar las características básicas de los dispositivos más comunes que se pueden utilizar para el diseño de sistemas de visión embebida y el consumo de potencia con respecto a su rendimiento en operaciones de coma flotante por segundo.

En cuanto al software, OpenCV es una biblioteca de visión por computadora de distribución libre y de código abierto que ofrece una amplia gama de funcionalidad bajo la licencia permisiva de Berkeley Software Distribution (BSD). La bibliotecaaestá escrito en C++ y también es utilizada a través de C o del lenguaje Python. Miles de desarrolladores utilizan OpenCV para alimentar sus propias aplicaciones especializadas, por lo que es la biblioteca más ampliamente utilizada. Un comunicado reciente también ha añadido soporte para la unidad de procesamiento gráfico (GPU) en la arquitectura de cálculo paralelo CUDA de NVIDIA.

Para muchos desarrolladores, OpenCV representa una atractiva caja de herramientas completa de algoritmos útiles, bien probados que puede servir como bloques para sus propias aplicaciones especializadas. La pregunta entonces es si o no OpenCV se puede utilizar directamente en sus sistemas embebidos. A pesar de su enfoque de desarrollo original para su uso con las estaciones de trabajo de PC, OpenCV también puede ser una herramienta útil para el desarrollo de sistemas de visión embebida. Hay bibliotecas en lenguaje C de varios proveedores que ofrecen capacidades de OpenCV como en varios sistemas embebidos, pero pocos pueden igualar la ubicuidad de OpenCV en el campo de la visión por computadora o la gran amplitud de sus algoritmos incluidos.

Tabla 3.1 Rendimiento del Hardware para procesamiento de imágenes.

Dispositivos	GFlops	Watts	características
Microcontrolador	<0.2	<0.3	La mayoría de los microcontroladores (por ejemplo: Arduino, AVR, PIC) son demasiado lentos para el procesamiento de la cámara, pero un ARM de 32 bits Cortex-M4 pueden manejar algunas aplicaciones muy básicas de la cámara tales como seguimiento y detección de color.
Targeta, movil, SOC	1-25	1-6	La mayoría de las tarjetas de propósito general hacen el procesamiento con cálculos de números enteros, y son de un precio accesible como el Cortex-A8 (BeagleBone Black) o incluso un ARM11 (Raspberry Pi) podría ser lo suficientemente bueno, pero si es necesario manejar operaciones con punto flotante, entonces definitivamente se necesita un Cortex-A9 o Cortex-A15. Pero si también es necesario visualizar imágenes en la pantalla entonces una tableta con Android o Linux podría ser una mejor opción. Actualmente la CPU ARM (Cortex A57/A72) de 32b/64b que usan en los dispositivos móviles pueden contener hasta 8 núcleos y proporcionan tanto una gran velocidad como un bajo consumo de energía
FPGA	50-1000	1-3	El procesamiento con FPGAs pueden ser extremadamente rápido con el mínimo de consumo en la batería, pero el modelo de programación es completamente diferente a la programación de software, aunque se puede insertar una CPU y un FPGA (sbRIO-9637, Nexys 3). La naturaleza flexible de la lógica programable y su estrecha integración con el sistema de procesamiento basado en ARM ofrece a los desarrolladores la posibilidad de añadir prácticamente cualquier periférico y crear aceleradores para ampliar el rendimiento de los dispositivos de visión embebida.

OpenCV ya ha sido portado a la arquitectura ARM, y es una opción muy popular para procesadores embebidos. Ciertamente es posible hacer una compilación cruzada usando el código de OpenCV, pero las limitaciones de memoria y otras consideraciones de la arquitectura de computadoras pueden suponer un problema. Un elemento muy importante que hay que considerar en el diseño de sistemas de visión

embebida es el compilador. El compilador GCC se ha utilizado en OpenCV con éxito para la plataformas ARM, pero GCC no está disponible en arquitecturas más especializados como los DSP. Estos dispositivos normalmente se basan en los compiladores propietarios que no son tan compatible con los estándares. Estos compiladores pueden tener un fuerte enfoque hacia el lenguaje C y ser menos capaces de optimizar el código C++. La versión actual de OpenCV se basa principalmente en C++ Standard Template Library (STL), así como del GCC y C99, que no están bien apoyados para ciertos compiladores de sistemas embebidos. Por estas razones, puede ser necesario volver a OpenCV versión 1.1 o anterior las cuales están escritos casi en su totalidad en C. El código fuente de OpenCV incluye muchas optimizaciones de bajo nivel para procesadores x86 que no son aplicables a las plataformas ARM o DSP. Estas optimizaciones se pueden reemplazar con las bibliotecas de apoyo proporcionados por el proveedor o funciones intrínsecas que hacen uso explícito de la arquitectura específica para manipular las instrucciones simples de múltiples datos (SIMD) y aceleraran la ejecución del código.

El uso de bibliotecas optimizadas debajo de las API en OpenCV pueden maximizar el rendimiento mediante la utilización de la arquitectura específica, manteniendo la interfaz estándar del software de alto nivel. En otras palabras, estas bibliotecas de bajo nivel pueden acelerar las funciones de OpenCV sin romper el código preexistente de la aplicación que esta escrito utilizando las API de OpenCV estándar. Otro de los retos que existe al usar funciones OpenCV en un entorno de procesamiento embebido es la falta de soporte nativo para operaciones matemáticas de puntos flotante. Esto plantea un problema significativo para usar OpenCV, ya que incluye una serie de funciones especializadas para el procesamiento de imágenes que dependen en gran medida del cálculo en puntos flotante. OpenCV es compatible con una amplia gama de tipos de datos de imagen, incluyendo representaciones de punto fijo y punto flotante. Sin embargo, algunas funciones especializadas para la transformación de la imagen siempre utilizan operaciones matemáticas de punto flotante, independientemente del original tipo de datos de imagen. Estos algoritmos intensivos requieren soporte nativo para conseguir un rendimiento en tiempo real en una aplicación embebida.

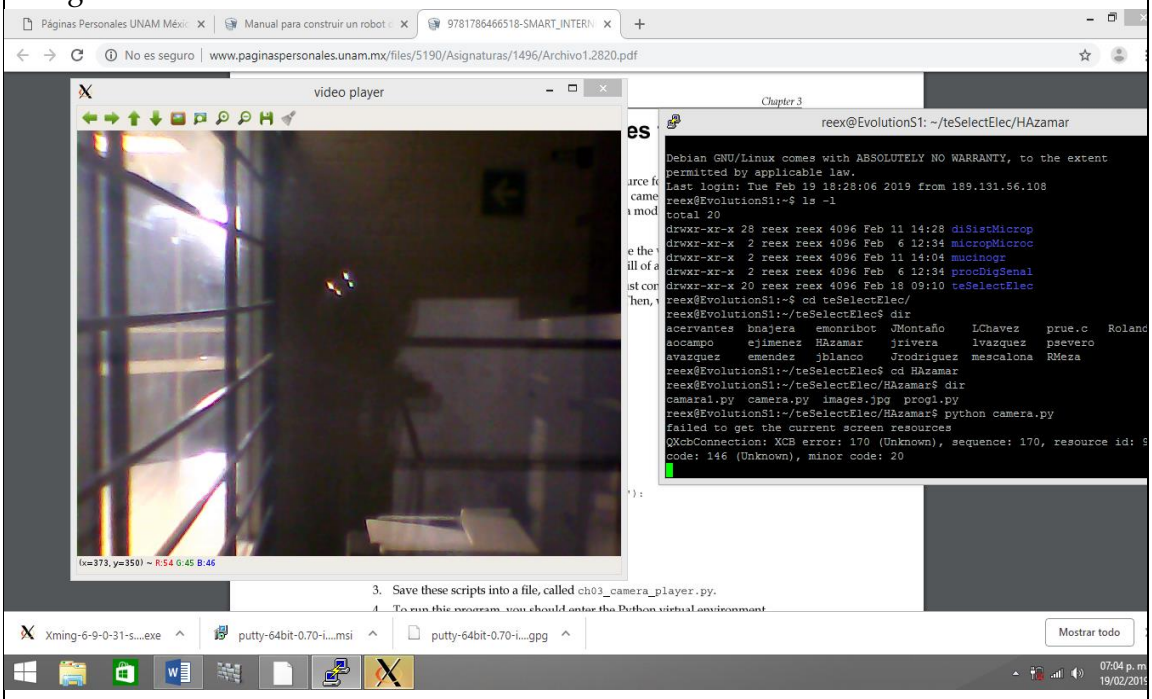
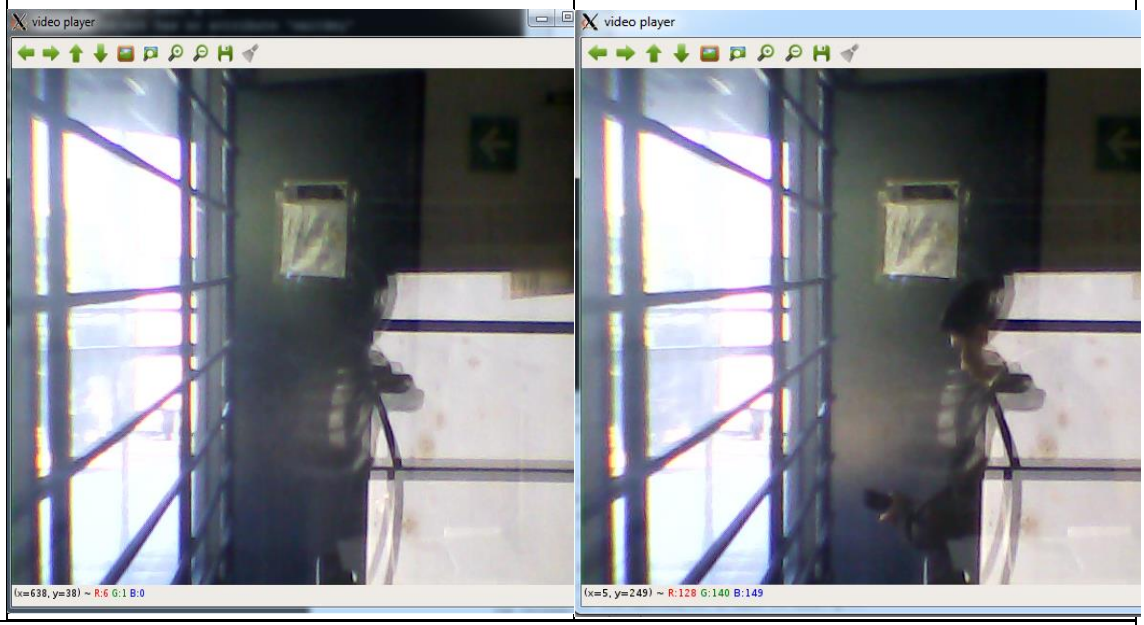
Con base en la información anterior y tomando en cuenta su bajo costo se puede elegir la tarjeta Raspberry Pi 3, con un procesador ARM11 convirtiéndose en una microcomputadora que ejecuta un sistema operativo Linux.

Ficha 3.8

Actividad 3.8.
Revisar el artículo: "PROTOTYPING AND SIMULATED ANALYSIS OF AUTONOMOUS SWARM-BOTS"
Maulikkumar Dhameliya, Sidharth Sher and Souma Chowdhury

ASME Proceedings | 42nd Mechanisms and Robotics Conference, 2018

Contribución Sistema de Visión embebida.

Resultados Experimento 3.1 Sem. 2019-II	imagen desde casa el 19 de febrero 2019 a las 7:04 
Detección Con Raspberry Pi Intruso/No Intruso	imágenes en la escuela lunes 18 de febrero 2019 

Actividad 3.9

a) Conectar a Internet la tarjeta Raspberry Pi.
b) Instalar OpenCV:

```
$ sudo su
# apt-get install update
# apt-get install build-essential git cmake pkg-config libgtk2.0-dev
# apt-get install libjpeg-dev libtiff5-dev
# apt-get install libjasper*
# apt-get install libpng*
# apt-get install libavcodec-dev libavformat-dev libswscale-dev
# apt-get install libv4l-dev
# apt-get install libxvidcore-dev libx264-dev
# apt-get install libatlas-base-dev gfortran
# apt-get install feh
# sudo apt-get install python-numpy
# sudo apt-get install python-scipy
# sudo apt-get install libopencv-*
```

c) Instalar en la PC Xming y PuTTy.
d) Acceder a la cuenta pi, a través de PuTTy

Ver material de clase "Archivos en Python para procesar imágenes". Tema 3.
Nota. Recuerde que hay que guardar el link con nombre .ZIP y descomprimir el archivo.

GPIO Raspberry Pi.

General Purpose Input Output (**GPIO**) es un sistema de entrada y salida de propósito general, es decir, consta de una serie de **pines o conexiones que se pueden usar como entradas o salidas** para múltiples usos. Estos pines están incluidos en todos los modelos de Raspberry Pi aunque con diferencias. Hay que tener en cuenta que dependiendo del modelo de la Raspberry Pi encontramos una cantidad de pines diferentes, por ejemplo, en la versión 1 de Raspberry Pi se tienen 26 pines GPIO mientras que a partir de la versión 2 de Raspberry Pi el número de pines aumentó a 40. Sin embargo, la compatibilidad es total, puesto que los 26 primeros pines mantienen su función original.

Los equipos de robots pequeños a menudo conocidos como swarm-bots, pueden proporcionar una funcionalidad única debido a su pequeño factor de forma, capacidades de detección distribuida, resiliencia a las interrupciones y pérdida de agentes, y bajo costo. Estos robots de enjambre se promocionan cada vez más para respaldar varias misiones de vigilancia en interiores, detección de peligros y de búsqueda y rescate.

Si bien existen varios interesantes swarm-bot y micro-bot. la mayoría de estos sistemas *i)* no pueden personalizarse fácilmente para aplicaciones de investigación específicas (más allá de para qué los crearon los desarrolladores originales; *ii)* muchos son simplemente herramientas educativas dadas allí movimiento lento y baja capacidad de control (por ejemplo, plataformas motorizadas por vibración); y *iii)* algunos se centran en la forma y el tamaño, para lograr un alcance operacional flexible en espacios reducidos.

Por tanto, este proyecto ha motivado el desarrollo de nuestra propia plataforma. El objetivo general de esta es apoyar de manera coherente nuestro enjambre haciendo implementaciones con los GPIO de la tarjeta Raspberry Pi.

Ficha 3.10

Actividad 3.10

Ver material de clase "GPIO Raspberry Pi". Tema 3.