UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE VISIÓN R.S. PARA CASCOS DE MOTOCICLETA

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO MECATRÓNICO

P R E S E N T A N:

HUGO RÓMAN ALDAMA GONZÁLEZ
Y
HÉCTOR ALEJANDRO RIVERA COLIN

DIRECTOR DE TESIS:
M.I BILLY ARTURO FLORES MEDERO NAVARRO
AÑO 2014

Ciudad Universitaria, D. F.

Índice

Índice de ilustraciones

Introducción

Desde hace unos años la implementación de los avances tecnológicos para el uso común se ha incrementado, es por ello que se muestra la propuesta de un casco con un retrovisor y que pronto podría ser inteligente.

También hay que tomar en cuenta que el tiempo que lleva para girar la cabeza y mirar hacia atrás, sobre un vehículo a alta velocidad, puede provocar una distracción que lleve al accidente.

El visor mostrado en esta tesis, tiene un cuerpo construido de MDF (tablero de fibra de densidad media), sobre el cual se monta una pantalla, una computadora *Mini PC Measy U2C*, una lente de 5 dioptrías y una mica reflejante semitransparente.

El sistema funciona tomando video continuo a través de la cámara que trae implementada la mini-computadora, mostrándolo en una pantalla. Este video pasa por la lente y se refleja en la mica haciéndolo llegar a nuestro ojo para, de esta manera, tener la imagen trasera y la frontal al mismo tiempo.

Esta tesis contiene 5 capítulos de los cuales:

En el capítulo 1 se habla de antecedentes históricos del casco para motocicleta, antecedentes en óptica geométrica, un poco de sistemas computacionales y diferencias entre los sistemas de visión que se han creado y/o descubierto.

En el capítulo 2 contiene toda la información referente a algunos modelos que se están realizando en otros países y como en la mayoría piensa en hacerlos inteligentes.

En el capítulo 3 se habla del desarrollo de la propuesta de sistema de visión RS para casco de motocicleta, aquí se muestran las propuestas iniciales, los detalles del problema a resolver y las propuestas con mejora computacional.

En el capítulo 4 se muestran los resultados obtenidos con cada una de las propuestas e incluso una comparación entre las mismas para observar cual es la mejor para este problema.

El capítulo 5 es un capítulo final y no por ello menos importante, aquí se muestran las conclusiones a las que se ha llegado, además de que se agregan trabajos a realizar en un futuro próximo para que el casco sea rentable.

Objetivos

Desarrollar un sistema de visión de realidad superpuesta (V.R.S.), que permita mostrar el conjunto de imágenes provenientes de la parte posterior de una motocicleta o de un casco, para ser visualizados por el usuario en la visera del mismo.

Justificación

El por qué del diseño un aditamento de visión.

Entre el 85 y el 90 por ciento de los accidentes de motocicletas se debe al factor humano.
Una colisión a 35 km/h equivale a una caída desde un cuarto piso.
Un motociclista que no use casco tiene 3 veces más probabilidad de sufrir un accidente mortal que un automovilista.
67 porciento de las lesiones mortales son de tipo craneal y cervical.
En este contexto, las tareas de concientización poblacional resultan de significativa trascendencia para despejar dudas y mitos acerca del uso del casco entre los motociclistas.

Argumentos refutables.

En cuanto a las aseveraciones de los conductores para justificar el no uso del casco se indica muy generalizadamente que provoca una "disminución de la visión". Sobre el particular, los estudios realizados han demostrado que ningún accidentado con casco lo ha sido por falta de visión; por el contrario, la visión otorgada por dicho elemento protector es de casi 180 grados, en tanto que esos accidentes analizados han ocurrido dentro de un campo visual – frontal de 120 grados. [1].

Estadísticas de accidentes de tránsito sin el uso del casco.

- Todos los años, los accidentes de tránsito vehicular causan 1,3 millones de muertes y entre 20 y 50 millones de heridos a nivel mundial.
- Los accidentes de tránsito se han convertido en la principal causa de muerte entre los jóvenes de 15 a 30 años.
- Estudios demuestran que si no se adoptan medidas, para el año 2020 los accidentes de tránsito causarán la muerte de 1,9 millones de personas por año.

- En varios países, las pérdidas causadas por accidentes de tránsito representan del 1 al 3% del Producto Interno Bruto.
- El 90% de los casos de traumatismo de cráneo y muerte por accidentes de tránsito ocurren en países pobres o en vías de desarrollo.
- En el 20% de los accidentes de moto el casco se desprende de la cabeza del usuario debido a que la medida del casco o la sujeción no fue la adecuada.
- Los traumatismos de cráneo son la lesión más frecuente y la principal causa de invalidez o muerte. Si la cabeza del conductor es protegida con un casco adecuado se pueden reducir estas lesiones en un 69% y la mortalidad un 42%.
- El riesgo de morir conduciendo una moto es 17 veces mayor que manejando un automóvil.[2]

También es importante usar la apropiada vestimenta para andar en moto y esto incluye: un buen par de guantes con agarre adecuado y relleno en los nudillos de los dedos, una chaqueta de motocicleta o un chaleco resistente, y pantalones para moto, como vaqueros o pantalones gruesos. Las botas para moto también son elementos de seguridad. [3]

Partiendo de las estadísticas anteriores esta tesis se propone para reducir el número de accidentes en motociclistas.

También se hace notar que la visión del conductor en una motocicleta tiene ciertos ángulos ciegos y que incluso cuando se tiene a un acompañante, esta visión es afectada.

Con la implementación de una cámara, el sistema óptico y la computadora se puede tener en cuenta que los puntos ciegos son eliminados, además de que la distracción directa al giro hacia los espejos se reduce puesto que ya no es necesario hacerlo una vez acostumbrado al sistema de visión.

Alcances

Construir un modelo funcional, el cual demuestre el principio de superposición de imágenes adentro de un casco, así como un sistema para reproducción de video.

El modelo funcional será construido de tal forma que el usuario lo pueda utilizar como un casco normal. Una de las soluciones es construir un visor con todos los elementos necesarios y montarlo en un casco ya existente.

La meta principal en el proyecto es de comprobar que el ser humano puede ver dos imágenes al mismo tiempo sin problema alguno, por lo tanto no es necesario que el casco cumpla con las normas de seguridad o que la imagen sea de alta resolución en esta primera etapa.

Capítulo 1 antecedentes

1.1 Historia del casco de motocicleta

La historia de la motocicleta empieza en 1885 cuando el Sr. Gottlieb Daimler fabricó su primer artefacto que era una rudimentaria moto con motor de combustión interna. Pero no fue hasta 1935 cuando alguien se planteó que el uso del casco sería beneficioso para los posibles accidentados. Ese alguien fue el neurocirujano Hugh Cairns, que quedó impactado al tener que asistir a la muerte de T.E. Lawrence (más conocido como Lawrence de Arabia, Figura 1) que había padecido un accidente a los mandos de su Brought Superior y probablemente se habría salvado de haber llevado esa prenda.

Figura 1: Lawrence de Arabia.

El Dr. Cairns utilizó para su estudio a los motociclistas del ejército inglés y ya en 1941 recomendó que los pilotos debieran llevar casco. En 1946 publicó un segundo estudio en el que certificaba que los motociclistas, que habían seguido la recomendación de 1941, habían sufrido menos heridas y de menor consideración que los que no utilizaban casco. No obstante la recomendación de Cairns no fue obligación hasta 1973 en Francia y Reino Unido, 21 años después de su muerte.

El primer casco de moto fue diseñado como un casco para la aviación. Desde que se presentó el primero, los organismos de seguridad, principalmente gubernamentales reconocieron la necesidad de protección de la cabeza para motociclistas. En la Figura 2 se aprecia el casco de F. Lombard quien agrego una protección acolchada en el interior.

Figura 2: Casco de F. Lombard.

En 1961, se aprobó la primera ley en el mundo que hacia obligatorio el uso del casco. En 1966, a lo largo de la controversia sobre las leyes de casco obligatorio, los fabricantes tuvieron que ofrecer mejor protección a los corredores. En 1967 se introdujo el primer casco completo, proporcionando una visión mejorada. Fue desarrollado como un casco más fuerte y más ligero, dando a los pilotos de motos mayor comodidad y mejor protección. En 1957 se formó *Snell Memorial Foundation* con el fin de proporcionar pruebas de casco de motocicleta independiente. En 1974 el Departamento de Transporte de Estados Unidos (DOT) introdujo las normas federales de seguridad de vehículos de motor para cascos de motocicletas. Las normas establecidas por DOT obligan a que los fabricantes de cascos colocaran una etiqueta de aprobación dentro de cada casco fabricado. [4].

También cabe señalar que para la segunda guerra mundial la mayoría de los motociclistas ya usaban casco incluso el propio militar como se ve en la figura 3.

Figura 3: Soldado con casco.

En la figura 4 se muestra un promocional de cascos.

Figura 4: Promocional de cascos para motocicleta.

1.2 Partes de los cascos de motocicletas

Las partes que componen un casco de motocicleta son:

Parte externa, parte inferior, poliexpan, pantalla y sistema de respiración.

Son las principales partes que lo componen debido a sus grandes cambios desde su creación misma, para tener mayor protección y comodidad que se acata conforme las normas de seguridad en cascos de motocicleta una de ellas es la DOT.

Figura 5: Partes del casco de motocicleta.

Como se muestra en la Figura 5, en las partes de la motocicleta hay una que se llama pantalla y esto solo es la protección contra el viento. Se piensa utilizar esta parte para reflejar la imagen que se desea, sin embargo por las características que tiene el material no es posible, de tal modo que se busca un material que haga la función.

Se realiza una propuesta para dar comodidad y que no existan molestias con los ojos y dolor de cabeza, se implementa un sistema óptico con lentes convergentes y se introduce una computadora (Mini Pc) para que la imagen deseada pueda ser procesada y con ello tener mejores resultados.

1.3 Lentes delgadas y tipos de imágenes producidas por estas

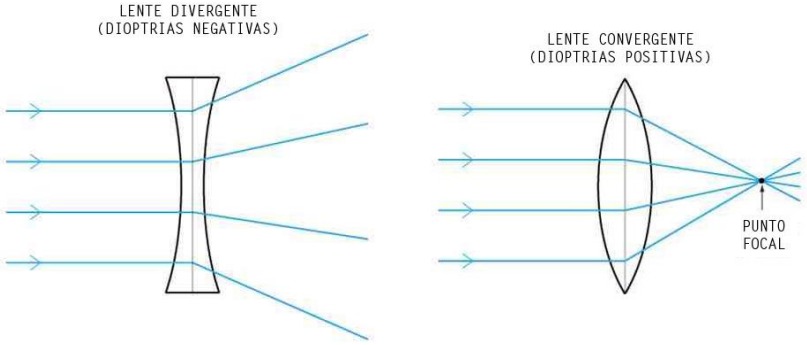


Figura 6: Tipos de Lentes.

Lentes Divergentes.

Las lentes divergentes son más delgadas en el centro que en los bordes. Su imagen focal es menor que 0. Todos los rayos paralelos que inciden sobre ella, salen divergiendo de la misma, de forma que parece que parten de un mismo punto anterior a la misma lente.

Pueden ser meniscos divergentes (conocidos como meniscos negativos), lentes plano - cóncavas y bicóncavas.

Estas lentes producen imágenes virtuales, directas y de menor tamaño. Particularmente, al situar el objeto sobre el foco imagen (F') obtenemos una imagen cuyo aumento lateral es la mitad del tamaño del objeto original.

Si miramos por una lente divergente da la sensación de que los rayos proceden del punto F. A éste punto se le llama foco virtual.

En las lentes divergentes como la de la figura 7 la distancia focal se considera negativa.

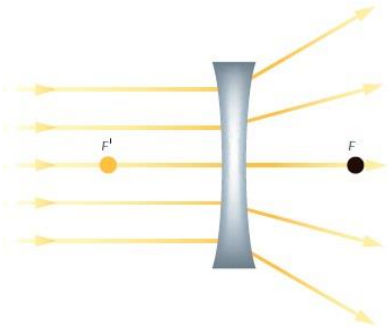


Figura 7: Lente divergente.

Lentes convergentes.

Una lente convergente o lente positiva es aquella en la que los rayos paralelos que inciden sobre la misma son desviados hacia el mismo punto.

Dioptrías

La dioptría es la unidad que expresa con valores positivos o negativos la potencia de una lente o poder de refracción de una lente, y equivale al valor reciproco de su longitud focal expresada en metros

Imágenes.

En óptica geométrica una imagen es una figura formada por el conjunto de puntos donde convergen los rayos que provienen de las fuentes puntuales del objeto tras su interacción con el sistema óptico, según este sea conformado de un espejo o lente, puede haber imágenes reales o virtuales.

Una imagen virtual es la imagen formada en la intersección de las prolongaciones de los rayos refractados o reflejados.

Una imagen real es la imagen formada por la intersección de los rayos reflectados o reflejados (son convergentes).

Una imagen real es aquella que se puede proyectar sobre una superficie para verla.

Una imagen virtual es aquella que aparenta estar por detrás de la lente, no puede ser proyectada, pero puede ser vista cuando se enfoca con los ojos.

1.4 Formación de imágenes en lentes delgadas

La distancia a la que la lente forma la imagen depende de la distancia focal propia de la lente. La distancia focal (F) de una lente convergente es la distancia entre el centro óptico de la lente y el foco cuando se enfoca al infinito, es decir cuando los rayos de luz provenientes del infinito convergen en un punto.

Las distancias del objeto a la lente(s) y la distancia entre la imagen formada a la lente(s'), se relacionan mediante la ecuación (1) de la cual se obtienen cuatro casos:

$$\frac{1}{\mathrm{F}} = \frac{1}{\mathrm{s}'} + \frac{1}{\mathrm{s}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots ec\ 1$$

Caso A.

El objeto real crea una imagen real invertida. Si el objeto está a una distancia 2F (donde F es la distancia focal) de la lente la imagen se creara a 2F de la lente, este caso es especial porque la proyección mantiene el tamaño del objeto, Figura 8.

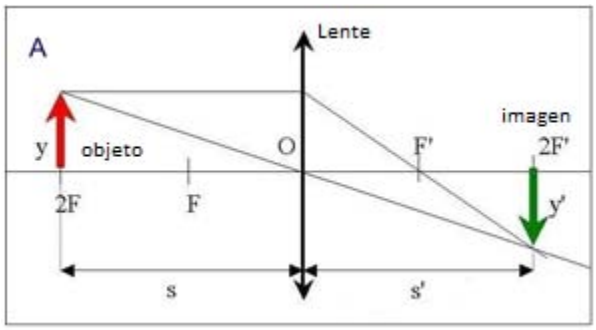


Figura 8: Caso A.

Caso B.

El objeto real crea una imagen real invertida pero la proyección magnifica el tamaño de la imagen con respecto al objeto solo si este se ubica a una distancia entre F y 2F, en cambio si el objeto se encuentra a una distancia mayor de 2F la imagen disminuirá su tamaño, Figura 9.

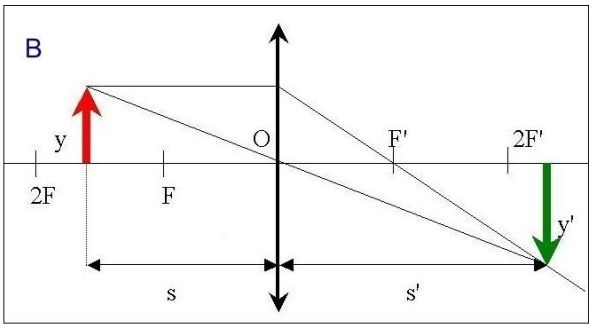


Figura 9: Caso B.

Caso C.

El objeto virtual crea una imagen real aumentada. Algo importante para que esto suceda es que el objeto debe haber sido creado por otra lente previamente, Figura 10.

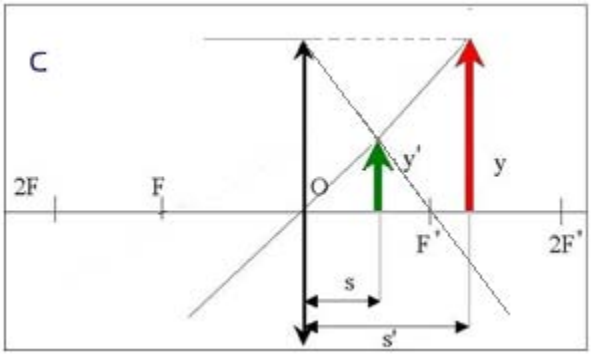


Figura 10: Caso C.

Caso D.

El objeto real crea una imagen virtual aumentada es necesario que la distancia entre el objeto y la lente sea menor a F.

Este último caso fue utilizado para el sistema de lentes que se usa en el casco, Figura 11.

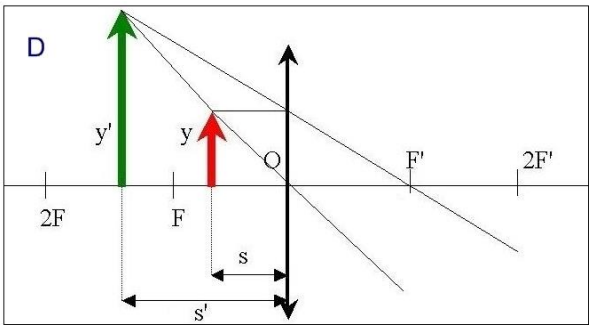


Figura 11: Caso D.

1.5 Divisor de luz

Un divisor de haz (o divisor de luz) es un instrumento óptico que divide un rayo de luz en dos. Es una parte fundamental de la mayoría de los interferómetros.

En su forma más común, un cubo, está compuesto por dos prismas de vidrio triangulares que están pegados por la base usando bálsamo de Canadá. El grosor de la capa de resina es ajustado para que —a una cierta longitud de onda— la mitad de la luz incidente en una "puerta" (es decir, una cara del cubo) sea reflejada y la otra mitad sea transmitida.

Otro diseño es mediante el uso de espejos semitransparentes. Consiste en una lámina de vidrio con un delgado revestimiento de aluminio (normalmente depositado mediante vapor de aluminio) con un grosor tal que, para un rayo de luz que incida con un ángulo de 45º, la mitad es transmitida y la otra mitad es reflejada. En lugar de un revestimiento metálico, también puede usarse un revestimiento óptico dieléctrico. Estos espejos se usan normalmente como acopladores de salida en la construcción de un láser.

Una tercera versión del divisor de haz es mediante el ensamblado de prismas dicroicos, que usa revestimientos ópticos dicroicos para dividir la luz entrante en tres rayos: rojo, verde y azul. Este dispositivo se usa en cámaras de televisión multitubo en color y en las Technicolor de tres películas. También se usa en los proyectores de tres LCD para separar los colores y en los focos reflectores elipsoidales para eliminar la radiación de calor.

Los divisores de haces también se usan en estereoscopía para hacer fotos estéreo utilizando un sólo disparo con una cámara no estéreo. El dispositivo se coloca en lugar del objetivo de la cámara.

1.6 El ojo humano

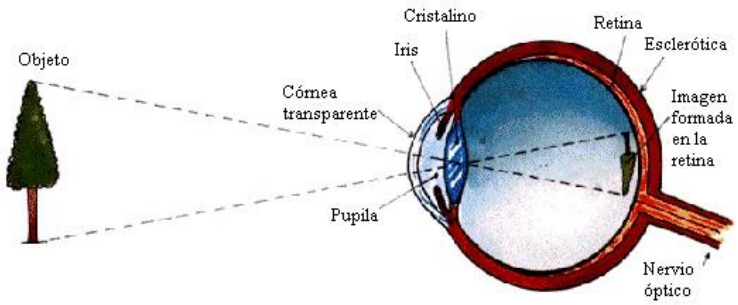


Figura 12: Ojo Humano.

El campo visual es el área dentro de la cual se perciben imágenes alrededor de un objeto determinado sobre el cual se mantiene la vista fija. El campo visual varía de persona a persona.

El ojo humano dispone de un campo visual. Cada ojo ve aproximadamente 150º sobre el plano horizontal y con la superposición de ambos se abarcan los 180º. Sobre el plano vertical sólo son unos 130º, 60º por encima de la horizontal y 70º por debajo como se observa en la figura 13.

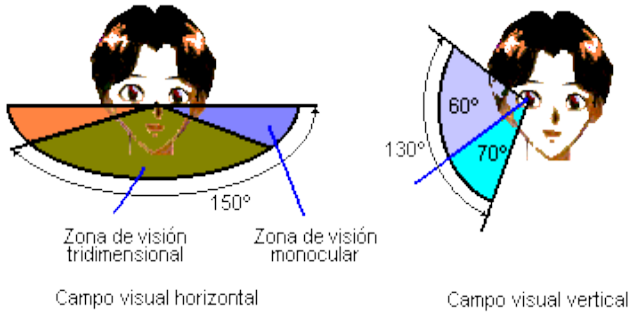


Figura 13: Campo visual del ojo humano.

El ojo sano y normal (Fig. 14) ve los objetos situados en el infinito sin acomodación (movimiento que realiza el cristalino para enfocar objetos en la retina). Esto quiere decir que el foco está en la retina y el llamado punto remoto (Pr) está en el infinito.

Se llama punto remoto la distancia máxima a la que puede estar situado un objeto para que una persona lo distinga claramente y punto próximo a la distancia mínima.

Un ojo normal figura x será el que tiene un punto próximo a una distancia "d" de 25 cm, (para un niño puede ser de 10 cm) y un punto remoto situado en el infinito. Si no cumple estos requisitos el ojo tiene algún defecto.

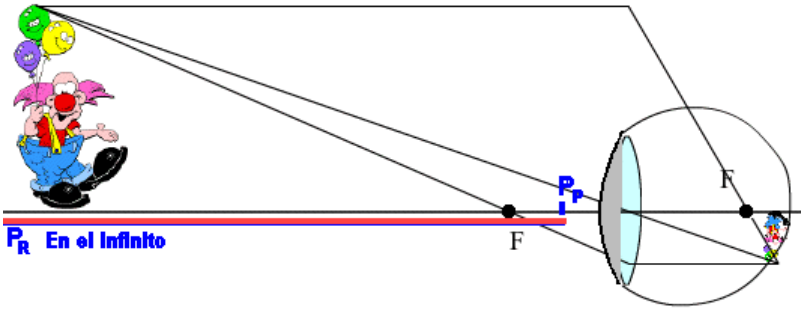


Figura 14: Ojo Sano.

Defectos De La Visión:

Miopía:

El ojo miope es un sistema óptico con un exceso de convergencia.

La persona miope (Fig. 15) no ve bien de lejos. Al estar el punto focal del ojo más cerca de la córnea que en un ojo normal, los objetos situados en el infinito forman la imagen delante de la retina y se ven borrosos. Empiezan a verse bien cuando están más cerca.

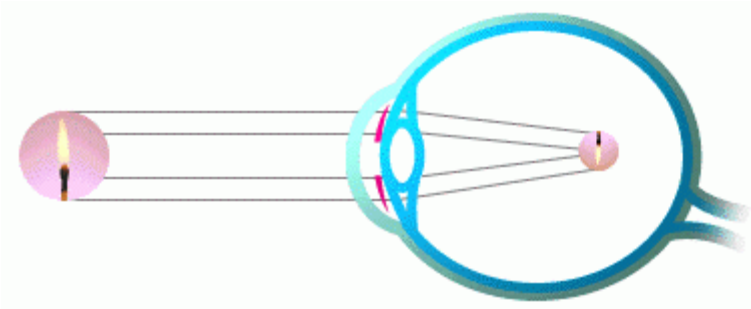


Figura 15: Ojo con miopía.

Para corregir la miopía se necesitan lentes divergentes para corregir los rayos que llegan como se observa en la figura 16.

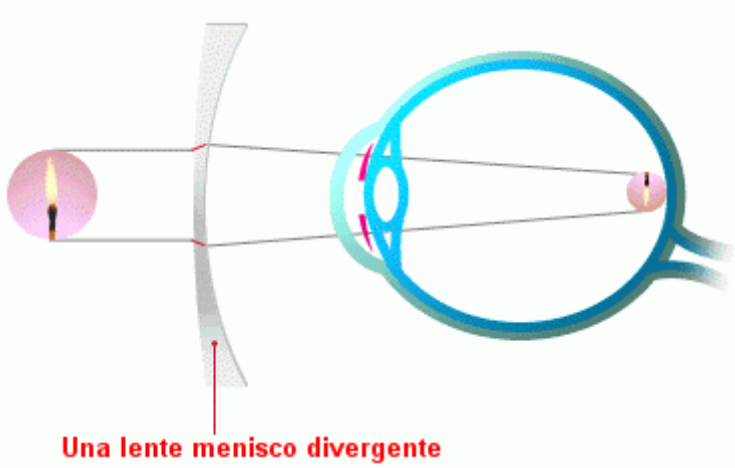


Figura 16: Ojo con miopía y lente divergente.

Hipermetropía:

Es una deficiencia de convergencia del sistema óptico del ojo (véase figura 17). El foco imagen del ojo está detrás de la retina cuando el ojo está en actitud de descanso sin empezar la acomodación.

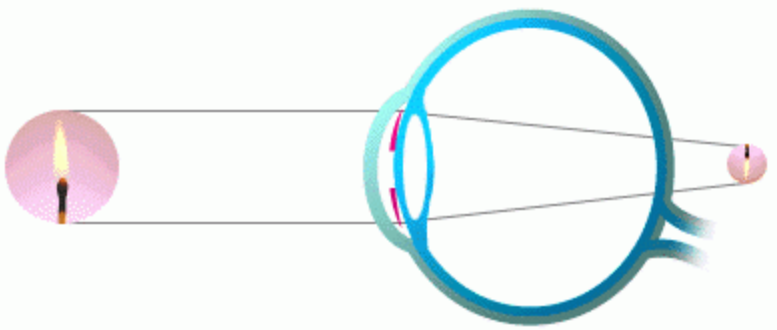


Figura 17: Ojo con hipermetropía.

La hipermetropía se corrige con lentes convergentes (véase figura 18). En algunos casos se corrige al crecer la persona y agrandarse el globo ocular.

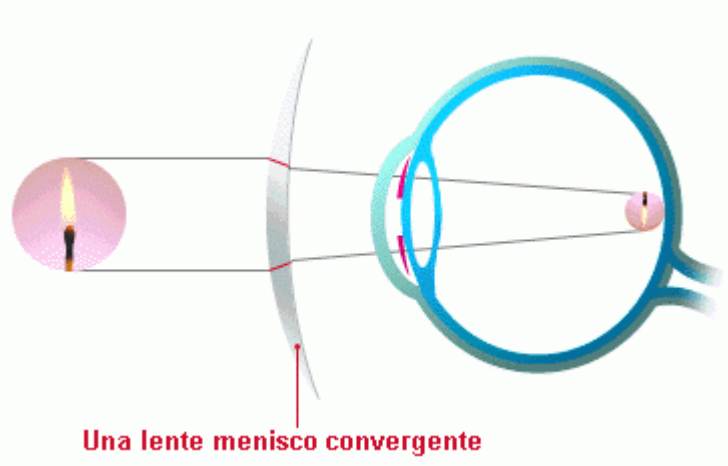


Figura 18: Solución al ojo con hipermetropía.

Vista cansada.

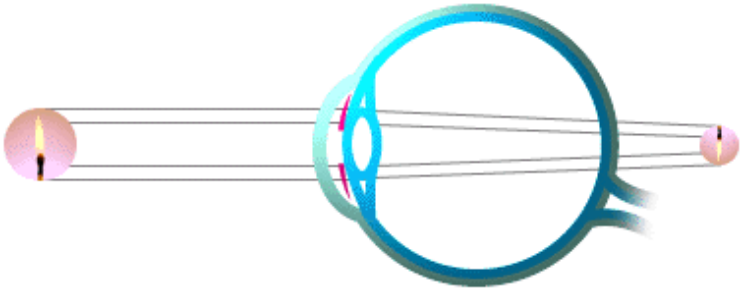


Figura 19: Ojo de Vista cansada.

Con el paso de los años se reduce la capacidad de adaptación del cristalino (pierde flexibilidad) y aumenta la distancia a la que se encuentra el punto próximo (véase figura 19). Este defecto se llama presbicia y se corrige con lentes convergentes (Fig 20).

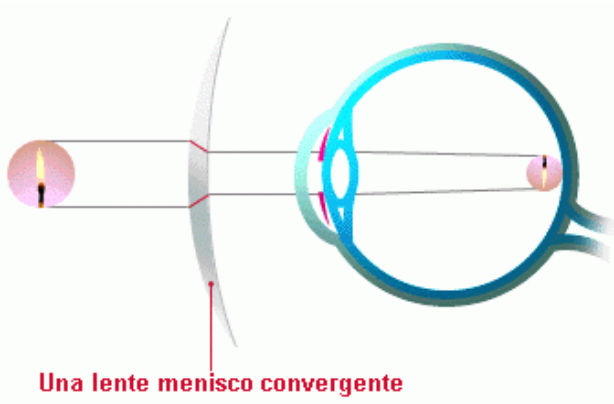


Figura 20: Solución al ojo de vista cansada.

Astigmatismo.

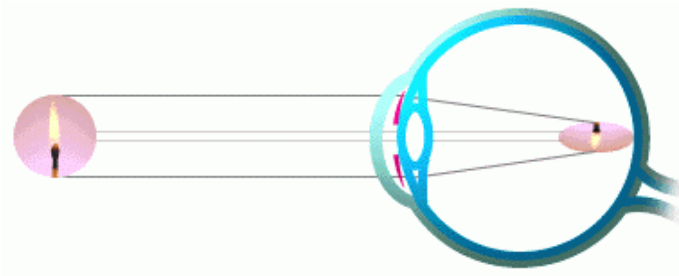


Figura 21: Ojo Astigmático.

Si el ojo tiene una córnea deformada (como si la córnea fuese esférica con una superficie cilíndrica superpuesta) los objetos puntuales dan como imágenes líneas cortas como se observa en la figura 21). Este defecto se llama astigmatismo y para corregirlo es necesaria una lente cilíndrica compensadora (figura 22).

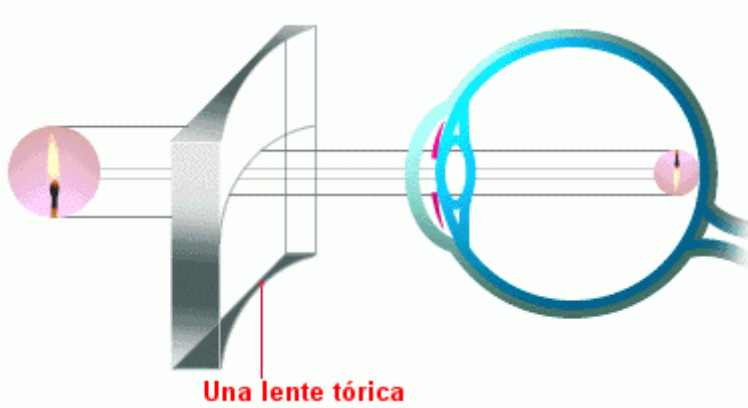


Figura 22: Solución al ojo con astigmatismo.

1.7 Arquitecturas de computadoras y procesadores

CPU o procesador, es el componente principal de la computadora y otros dispositivos programables, que interpreta las instrucciones contenidas en los programas y procesa los datos. Las CPU proporcionan la característica fundamental del ordenador digital (la forma de programar) y son uno de los componentes necesarios encontrados en las computadoras de cualquier tipo, junto con la memoria principal y los dispositivos de entrada/salida. Se conoce como microprocesador al CPU que es manufacturado con circuitos integrados. Desde mediados de los años 1970, los microprocesadores de un solo chip han reemplazado casi totalmente todos los tipos de CPU y hoy en día, el término "CPU" es aplicado usualmente a todos los microprocesadores. La expresión "unidad central de proceso" es, en términos generales, un dispositivo lógico que pueden ejecutar complejos programas de ordenador. Esta amplia definición puede fácilmente ser aplicada a muchos de los primeros ordenadores que existieron mucho antes que el término "CPU" estuviera en amplio uso. Sin embargo, el término en sí mismo y su acrónimo han estado

en uso en la industria de la Informática por lo menos desde el principio de los años 60. La forma, el diseño y la implementación de las CPU ha cambiado drásticamente desde los primeros ejemplos, pero su operación fundamental ha permanecido casi intacta.

1.7.1 Arquitecturas de procesador o CPU

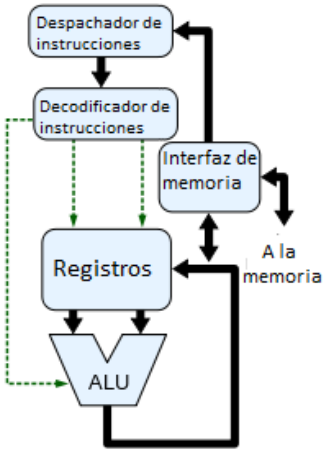


Figura 23: Diagrama de bloques de un CPU simple.

Arquitectura CISC (del inglés *Complex Instruction Set Computer*, en español Computadora con Conjunto de Instrucciones Complejas).

Los microprocesadores CISC tienen un conjunto de instrucciones que se caracteriza por ser muy amplio y permitir operaciones complejas entre operandos situados en la memoria o en los registros internos, en contraposición a la arquitectura RISC.

Este tipo de arquitectura dificulta el paralelismo entre instrucciones, por lo que, en la actualidad, la mayoría de los sistemas CISC de alto rendimiento implementan un sistema que convierte dichas instrucciones complejas en varias instrucciones simples del tipo RISC, llamadas generalmente microinstrucciones.

Los CISC pertenecen a la primera corriente de construcción de procesadores, antes del desarrollo de los RISC. Ejemplos de ellos son: Motorola 68000, Zilog Z80 y toda la familia Intel x86, AMD x86-64 usada en la mayoría de las computadoras personales actuales.

Hay que hacer notar, sin embargo que la utilización del término CISC comenzó tras la aparición de los procesadores RISC como nomenclatura despectiva por parte de los defensores/creadores de estos últimos.

SISC (*Simple Instruction Set Computing*) es un tipo de arquitectura de microprocesadores orientada al procesamiento de tareas en paralelo. Esto se implementa mediante el uso de la tecnología VLSI (*Very Large Scale Integration*), que permite a múltiples dispositivos de bajo costo que se utilicen conjuntamente para resolver un problema particular dividido en partes disjuntas. La arquitectura RISC es un subconjunto del SISC, centrada en la velocidad de procesamiento debido a un conjunto de instrucciones reducido.

RISC (*Reduced Instruction Set Computer*), en español Computadora con Conjunto de Instrucciones Reducidas, es un tipo de diseño de CPU generalmente utilizado en microprocesadores o microcontroladores con las siguientes características fundamentales:

- Instrucciones de tamaños fijos y presentados en un reducido número de formatos.
- Sólo las instrucciones de carga y almacenamiento acceden a la memoria de datos.

Además estos procesadores suelen disponer de muchos registros de propósito general. El objetivo de diseñar máquinas con esta arquitectura es posibilitar la segmentación y el paralelismo en la ejecución de instrucciones y reducir los accesos a memoria. Las máquinas RISC protagonizan la tendencia actual de construcción de microprocesadores. PowerPC, DECAlpha, MIPS, ARM, SPARC son ejemplos de algunos de ellos.

RISC es una filosofía de diseño de CPU para computadora que está a favor de conjuntos de instrucciones pequeñas y simples que toman menor tiempo para ejecutarse.

El tipo de procesador más comúnmente utilizado en equipos de escritorio es X86, está basado en CISC en lugar de RISC, aunque las versiones más nuevas traducen instrucciones basadas en CISC x86 a instrucciones más simples basadas en RISC para uso interno antes de su ejecución.

La idea fue inspirada por el hecho de que muchas de las características que eran incluidas en los diseños tradicionales de CPU para aumentar la velocidad estaban siendo ignoradas por los programas que eran ejecutados en ellas. Además, la velocidad del procesador en relación con la memoria de la computadora que accedía era cada vez más alta. Esto conllevó la aparición de numerosas técnicas para reducir el procesamiento dentro del CPU, así como de reducir el número total de accesos a memoria.

Terminología más moderna se refiere a estos diseños como arquitecturas de carga-almacenamiento.

1.7.2 Arquitectura ARM

ARM es una arquitectura RISC (*Reduced Instruction Set Computer* que significa Computadora con Conjunto de Instrucciones Reducidas) de 32 bits desarrollada por ARM Holdings. Se llamó *Advanced RISC Machine*, y anteriormente *Acorn RISC Machine*. La arquitectura ARM es el conjunto de 32 bits más ampliamente utilizado en unidades producidas. Concebida originalmente por *AcornComputers* para su uso en computadoras personales, los primeros productos basados en ARM eran los *AcornArchimedes*, lanzados en 1987.

La relativa simplicidad de los procesadores ARM los hace ideales para aplicaciones de baja potencia. Como resultado, se han convertido en dominante en el mercado de la electrónica móvil e integrada, encarnados en microprocesadores y microcontroladores pequeños, de bajo consumo y relativamente bajo coste. En 2005, alrededor del 98% de los más de mil millones de teléfonos móvilesvendidos cada año utilizan al menos un procesador ARM. Desde 2009, los procesadores ARM son aproximadamente el 90% de todos los procesadores RISC de 32 bits integradosy se utilizan ampliamente en la electrónica de consumo, incluyendo PDA, tabletas, teléfonos inteligentes, teléfonos móviles, videoconsolas portátiles, calculadoras, reproductores digitales de música y medios (fotos, vídeos, etc.), y periféricos de ordenador como discos duros y Routers.

1.7.3 Arquitecturas de computadoras

Arquitectura Von Neumann.

La arquitectura de Von Neumann (véase figura 24) es una familia de arquitecturas de computadoras que utilizan el mismo dispositivo de almacenamiento tanto para las instrucciones como para los datos.

La mayoría de computadoras modernas están basadas en esta arquitectura, aunque pueden incluir otros dispositivos adicionales, (por ejemplo, para gestionar las interrupciones de dispositivos externos como ratón, teclado, etc).

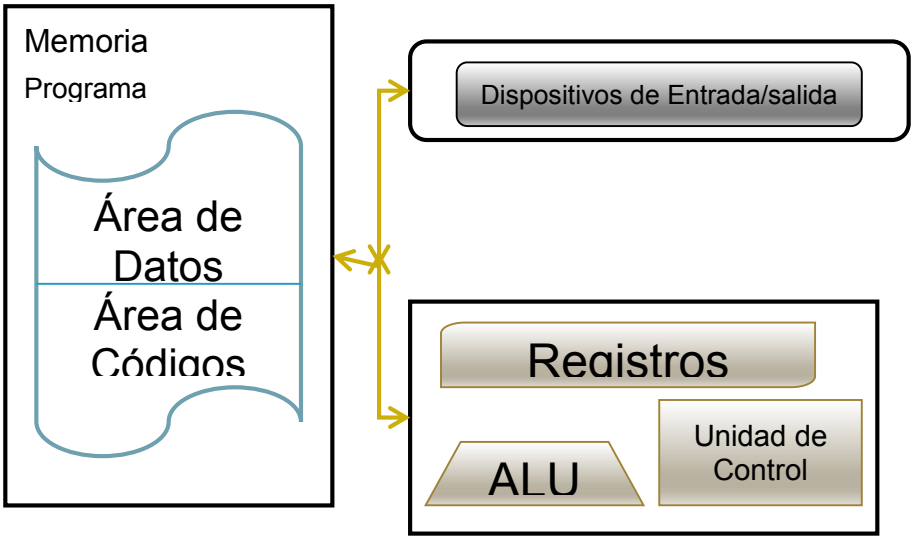


Figura 24: Diagrama arquitectura Von Neumman.

Características principales:

- Los datos y los programas se almacenan en la memoria y son gestionados por el mismo sistema de manejo de información.
- En un equipo que utiliza la arquitectura Von Neumann, sin caché, el CPU puede ser la lectura/instrucción/escritura, es decir, ambas operaciones no pueden realizarse simultáneamente.

Arquitectura Harvard.

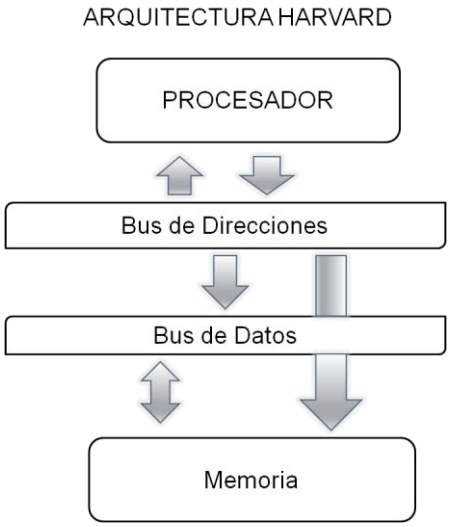


Figura 25: Diagrama arquitectura Harvard.

Originalmente, el término Arquitectura Harvard (véase figura 25) hacía referencia a las arquitecturas de computadoras que utilizaban dispositivos de almacenamiento físicamente separados para las instrucciones y para los datos (en oposición a la Arquitectura de Von Neumann). El término proviene de la computadora Harvard Mark I, que almacenaba las instrucciones en cintas perforadas y los datos en interruptores.

Las instrucciones y los datos se almacenan en cachés separadas para mejorar el rendimiento. Por otro lado, tiene el inconveniente de tener que dividir la cantidad de caché entre los dos, por lo que funciona mejor sólo cuando la frecuencia de lectura de instrucciones y de datos es aproximadamente la misma.

Características principales:

Los datos y programas se almacenan en dispositivos de memoria independientes y manejados por diferentes subsistemas.

En la arquitectura Harvard el CPU puede dar una instrucción y los datos de acceso a la memoria en el mismo tiempo sin memoria caché.

1.8 PIC

Un microcontrolador (abreviado µC, UC o MCU) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.

Debido a sus módulos periféricos tales como timers, convertidor analógico digital, PWM, módulos de comunicación (I2C, UART, MSI, etc), comparadores analógicos, puede usarse en un sinnúmero de aplicaciones que requieran un control eléctrico.

Los PIC son una familia de microcontroladores tipo RISC fabricados por Microchip Technology Inc. y derivados del PIC1650, originalmente desarrollado por la división de microelectrónica de *General Instrument*.

1.8.1 I2C

I2C es un bus de comunicaciones en serie. Es un bus muy usado en la industria, principalmente para comunicar microcontroladores y sus periféricos en sistemas integrados.

1.9 Raspberry Pi

Raspberry Pi es un ordenador de placa reducida (SBC, Fig. 27, 28) de bajo costo desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

El diseño incluye un *System-on-a-chip* Broadcom BCM2835, que contiene un procesador central (CPU) ARM1176JZF-S a 700 MHz (el *firmware* incluye unos modos "Turbo" para que el usuario pueda hacerle *overclock* de hasta 1 GHz sin perder la garantía), un procesador gráfico (GPU) VideoCore IV, y 512 MB de memoria RAM aunque originalmente al ser lanzado eran 256 MB. El diseño no incluye un disco duro o una unidad de estado sólido, ya que usa una tarjeta SD para el almacenamiento permanente; tampoco incluye fuente de alimentación o carcasa. El modelo B se vende a 35 USD y el modelo A a 25 USD. El 29 de febrero de 2012 la fundación empezó a aceptar órdenes de compra del modelo B, y el 4 de febrero de 2013 del modelo A.

La fundación da soporte para las descargas de las distribuciones para arquitectura ARM (véase Figura 26), Raspbian (derivada de Debian), RISC OS5, Arch Linux ARM (derivado de Arch Linux) y Pidora (derivado de Fedora).

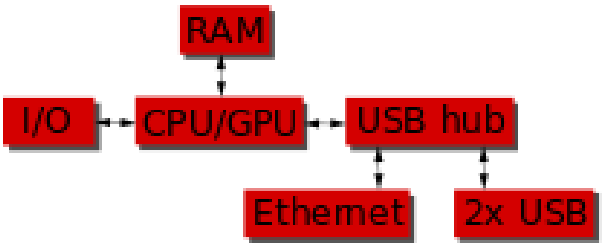


Figura 26: Diagrama de arquitectura ARM de Raspberry Pi.

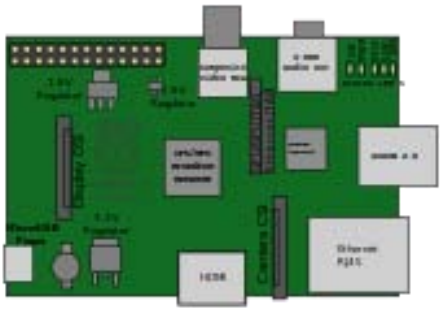


Figura 27: Placa de la Raspberry Pi.

Figura 28: Raspberry Pi funcionando.

1.10 Mini PC

Figura 29: Mini PC.

Un Mini PC (Fig. 29) es una computadora de tamaño y forma similares a una barra USB, utiliza Android como sistema operativo, aunque algunos modelos también funcionan con distribuciones de Linux como picuntu, tienen como procesador basado en AEM generalmente con núcleos *cortex*.

Tienen un puerto HDMI, y 2 USB, algunos periféricos extra como salida AV, entrada de tarjeta MicroSD, cámara web, micrófono, antena wifi y/o bluetooth.

El dispositivo se conecta a la televisión o pantalla mediante el puerto HDMI y tiene puertos USB para conectar dispositivos de interfaz humana como teclados, ratón, etc. Convirtiéndose así en una computadora de escritorio con la ventaja de ser portable ya que se puede conectar a cualquier pantalla o televisor con entrada HDMI.

Esta mini PC tiene el chip RK3066 que contiene varios periféricos como se muestra en la figura 30.

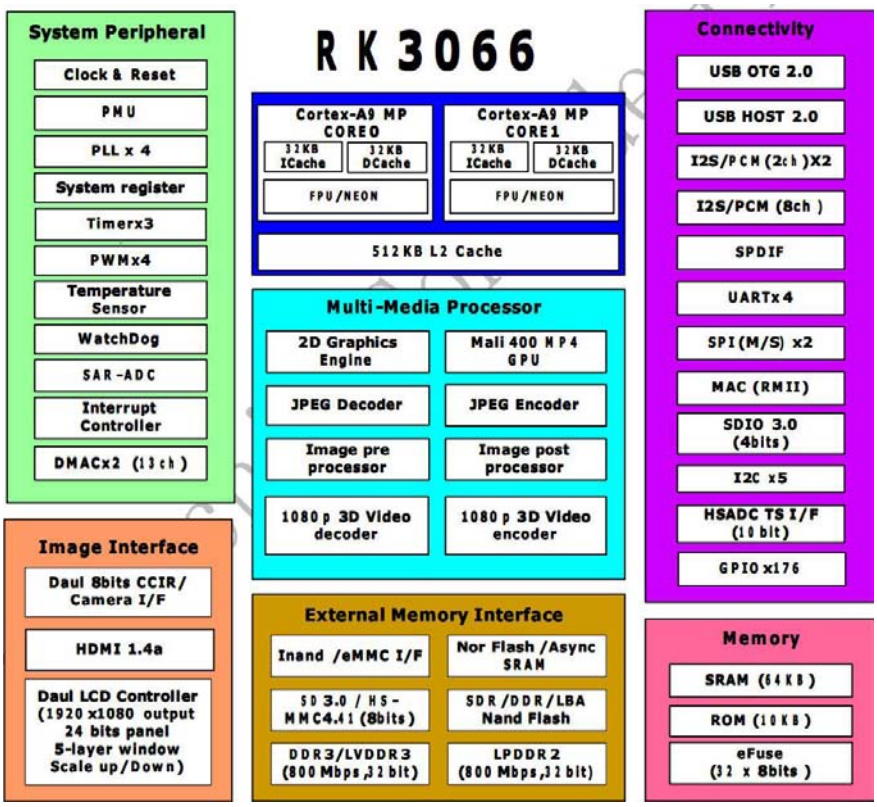


Figura 30: Diagrama de componentes de RK3066.

El chipRK3066, tiene el sistema operativo Android 4.1.1 y sus características generales son:

- Tecnología de 40 nm.
- Dos núcleos Cortex-A9 (ARMv7) hasta 1.6 GHz.
- Soporte para NEON (set de instrucciones para aceleración multimedia).
- Gpu Mali 400 Quad Core, a frecuencia de 250 MHz con soporte para Open GL ES 2.0 y Open VG 1.1 procesador multimedia VPU con soporte para la decodificación de imagen y video a 1080p.
- Soporte para memoria RAM tipo DDR, DDR2 y DDR3, hasta 2 GB.
- Interfaz HDMI 1.4 con 2 canales TFT LCD y soporte 3D (hasta 1920×1080).
- Interfaz USB 2.0 y SD/MMC.

1.11 Sistemas de visión

1.11.1 Realidad aumentada

La realidad aumentada (RA) es el término que se usa para definir una visión directa o indirecta de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real. Consiste en un conjunto de dispositivos que añaden información virtual a la información física ya existente, es decir, añadir una parte sintética virtual a lo real. Esta es la principal diferencia con la realidad virtual, puesto que no sustituye la realidad física, sino que sobreimprime los datos informáticos al mundo real.

Con la ayuda de la tecnología (por ejemplo, añadiendo la visión por computador y reconocimiento de objetos) la información sobre el mundo real alrededor del usuario se convierte en interactiva y digital. La información artificial sobre el medio ambiente y los objetos pueden ser almacenados y recuperados como una capa de información en la parte superior de la visión del mundo real.

La realidad aumentada de investigación explora la aplicación de imágenes generadas por ordenador en tiempo real a secuencias de vídeo como una forma de ampliar el mundo real. La investigación incluye el uso de pantallas colocadas en la cabeza, un *display* virtual colocado en la retina para mejorar la visualización, y la construcción de ambientes controlados a partir de sensores y actuadores.

Los elementos básicos que conforman un sistema de RA son los siguientes: un monitor, una cámara, el software ex profeso y un marcador. El marcador es un símbolo escrito o impreso sobre objetos determinados, que varía desde un código de barras de cualquier producto hasta datos codificados en la superficie de monumentos o edificios.

La cámara se encarga de captar esos símbolos y transferirlos al software. Éste interpreta los datos de los marcadores captados por la cámara y los convierte en todo tipo de información: texto, imágenes fijas, video en 3D o sonido.

Hasta ahora se tiene la realidad aumentada en algunos celulares inteligentes y las aplicaciones que desarrolladores han ido evolucionando.

Un ejemplo de ello se muestra en la figura 31.

Figura 31: Teléfono con aplicación de realidad aumentada.

Realidad aumentada en el futuro.

Se planea crear lentes de contacto que puedan proyectar imágenes directamente a nuestros ojos (Fig. 32). Hasta hace un tiempo, los lentes de contacto no eran más que una excelente opción para corregir problemas a la vista.

Figura 32: Ojo con lente de Realidad Aumentada.

1.11.2 Realidad virtual

La realidad virtual se podría definir como un sistema informático que genera en tiempo real representaciones de la realidad.

Se consigue mediante la generación por ordenador de un conjunto de imágenes que son contempladas por el usuario a través de un dispositivo denominado visor. Algunos de estos pueden ser artefactos colocados en un casco, anteojos (gafas) y hasta equipos integrados por trajes y guantes equipados con sensores diseñados para simular la percepción de diferentes estímulos, mismos que intensifican la sensación de realidad. Su aplicación, aunque centrada inicialmente en el terreno de los videojuegos, se ha extendido a otros muchos campos, como la medicina o las simulaciones de vuelo.

1.11.3 Realidad superpuesta

El término de realidad superpuesta no ha sido empleado en la actualidad, por ello se propone como definición de realidad superpuesta cuando en el mismo campo visual una persona es capaz de ver o de percibir dos imágenes provenientes de su entorno físico real, desde diferentes ángulos y que pueden o no ser superpuestas parcial o completamente.

A diferencia de la realidad aumentada que proporciona información adicional de un solo objeto, la realidad superpuesta proporciona información acerca de distintos entornos.

La realidad superpuesta tiene el objetivo de expandir la cantidad de información que es útil para el trabajo que realiza el usuario de este sistema.

Para esto se necesita una superficie que tenga las propiedades ópticas de transparencia y reflexión, de tal manera que se haga posible la superposición de imágenes.

Capítulo 2 Estado del Arte

2.1 Cascos Militares

La evolución de los cascos para motocicleta deriva de otros cascos como el de los pilotos de aviones y/o helicópteros, debido a que las mejoras en la visión son más aplicadas al área militar

2.1.1 BAE Systems

El nuevo casco para pilotos de cazadores Tornado probado en Afganistán.

BAE Systems (figura 33) ha entregado nuevos equipos a la Real Fuerza Aérea (RAF) como parte de un requerimiento operacional urgente para ayudar a los pilotos de aviones de combate Tornado a identificar fuerzas amigas. El equipo entregado es un nuevo casco con alta tecnología integrada que proyecta información delante de un ojo del piloto, permitiendo una evaluación instantánea de los puntos de interés. Esta innovadora tecnología puede ahorrar segundos vitales, así como vidas humanas, en prácticas operacionales de guerra.

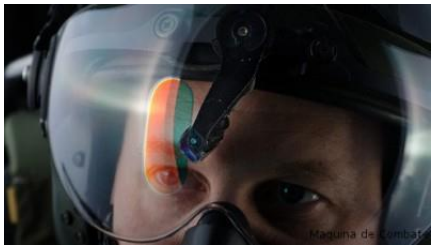


Figura 33: Piloto con casco BAE.

2.1.2 Casco de pilotos del F-35

Este casco permite a los pilotos del F-35 ver 'a través' del avión como se muestra en la figura 34.

Figura 34: Casco del F-35.

El casco trata de un nuevo ingenio diseñado para proporcionar a los pilotos de caza una mejor visión de sus alrededores. Detrás de ese particular visor, se encuentran dos proyectores sincronizados con varias cámaras instaladas en el avión, que muestran en directo lo que sucede alrededor del mismo. Esencialmente, el invento permite visualizar imágenes en alta resolución incluso de noche de áreas de otra forma invisibles a bordo de una apretada cabina, mientras que un conjunto de sensores integrados se encargan de decirle al operador hacia dónde está mirando en todo momento. Adicionalmente, los sistemas de a bordo son capaces de facilitarle "datos esenciales de vuelo y combate sobre la pantalla", como por ejemplo los símbolos para identificar a amigos y enemigos.

Este casco es tan sólo una de las muchas maravillas que incorporará el largamente esperado (y sofisticado) Joint Strike Fighter, por el cual la armada británica pagará 66 millones de libras (94 millones de euros/138 millones de dólares) una vez que entre en producción.

2.1.3 HEaDS-UP

Con el avance de la tecnología las fuerzas armadas cada vez más tratan de implementar las innovaciones en recursos para garantizar la seguridad de la gente.

En esta ocasión el ejército estadounidense está desarrollando un prototipo de casco que tiene un panel de 9 milímetros de protección en el cual se podrían proyectar imágenes usando un Smartphone Android (figura 35).

De acuerdo con el sitio de la armada de Estados Unidos, el nombre del dispositivo es *Helmet Electronics and Display System-Upgradeable Protection* (HEaDS-UP) y podría proyectar mapas estratégicos, localizaciones mediante GPS, métodos de comunicación en directo e incluso algunas aplicaciones comunes.

Figura 35: Casco HEaDS-UP.

2.1.4 JHMCS

Figura 36: Casco JHMCS.

El sistema de señalización montado en casco (JHMCS por sus siglas en inglés, figura 36) combina un rastreador de cabeza magnético con una pantalla proyectada en el visor del piloto, dando al piloto posición de los objetivos que puede ser usado para apuntar sensores y armas hacia donde el piloto mire.

Con el JHMCS, el piloto puede dirigir el radar, misiles aire a aire, sensores infrarrojos y armas aire a tierra meramente observando el objetivo a través del visor del casco y presionando un switch en los controles de vuelo. Adicionalmente el piloto puede ver cualquier dato deseado (velocidad, altitud, rango del objetivo, etc.) mientras ve hacia arriba, eliminando la necesidad de mirar a la cabina durante combate aéreo.

Fue implementado en el F-15, F-16 y F-22. Las pruebas de vuelo empezaron en el 2001.

2.2 Casco Reevu MSX1

El casco REEVU MSX1 es el sistema retro-visor está formado por varios espejos integrados en la caleta del casco. La visión posterior la capta una ventana transparente situada a la altura de nuestra coronilla y llega a un pequeño espejo retrovisor que queda por encima de nuestros ojos. De esta forma, echando un rápido vistazo hacia ese retrovisor veremos que sucede a nuestras espaldas (véase figura 37).

La visión que se consigue debe ser como la de la figura 38.

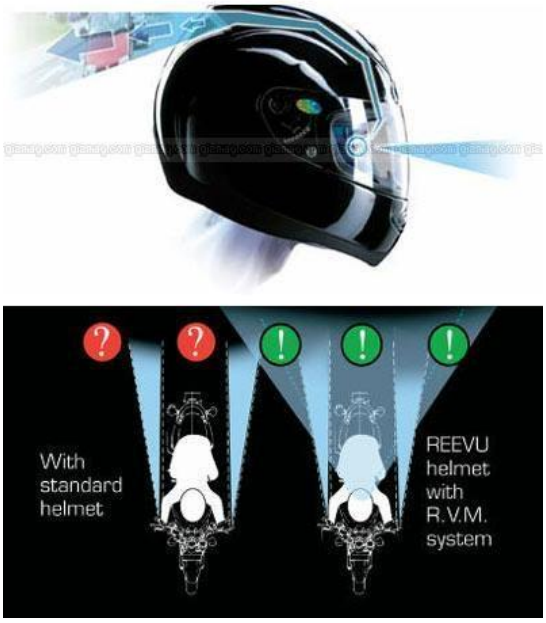


Figura 37: Visión ideal del casco Reevu MSX1.

Lógicamente el material de los espejos que forman el sistema no es vidrio, es una tela de policarbonato casi imposible de romper y muy ligera. El diseño del sistema de visión posterior permite que la estructura del casco sea completamente normal. Y al estar fabricado en materiales nobles, como el kevlar y el carbono, se trata de un producto de gran calidad.


Figura 38: Visión real del casco Reevu MSX1.

Pero no todo son ventajas. Algún defecto debía tener y, en este caso, son tres. Si circula con pasajero se verá su cara cada vez que eche un vistazo al retrovisor.

El segundo problema es que al circular muy acoplado en una deportiva la visión será la del cielo que se tiene a espaldas, cosa que por otro lado, viene bien para ver si te sigue el

helicóptero y un último defecto al usar lentes, la visión del retrovisor quedará anulada, ya que al subir la vista se verá la montura de las mismas y si se coloca para ver el espejo y si se tienen algunas dioptrías se verá borroso.

2.3 Casco Ruso LiveMap

Figura 39: Casco ideal LiveMap.

Innovando con el elemento principal para la protección de los motociclistas, se ha diseñado un casco que posee un sistema de navegación y georreferenciación para agilizar la movilidad y permitir a la persona que lo lleva, un despliegue de información sobre distancias, mapas y velocidad, que le facilita el desplazamiento en su medio de transporte.

LiveMap (figura 39) se convierte en la nueva herramienta tecnológica para los motociclistas, quienes tendrán delante de sus ojos, acceso a respuestas sobre el clima, tráfico y lugares recomendados, sitios de interés a la altura que se desplace. El dispositivo contiene un micrófono con el que se puede controlar a través de comandos de voz y un audífono para escuchar notificaciones y alertas. Para brindar mayor seguridad a la hora de conducir, los creadores rusos de este casco inteligente, incluyeron un método para bloquear funciones de videos o fotos mientras la persona se traslada de un lugar a otro.

Para ello el caso lleva incorporado un sistema de navegación GPS y un proyector semejante a los HUD ya vistos en los cascos de los pilotos de helicóptero. Para que su funcionalidad sea total y que el motociclista no tenga que utilizar sus manos en otra cosa que no sea la conducción de la motocicleta, el innovador casco llevará un sistema de reconocimiento de voz basado en la tecnología Siri de Apple. De este modo se puede consultar directamente dónde se encuentra la próxima gasolinera, algún hotel, o si hay retenciones en la carretera.

Toda esta tecnología no supondrá ninguna merma en la seguridad pasiva pues está homologado según los principales estándares mundiales (Europa ECE 22.05, USA DOT, y japonesa JIS T8133). Algo que se consigue, en parte, gracias a su estructura en fibra de carbono. Un material imprescindible para aligerar un conjunto más grande de lo normal para alojar todos los dispositivos extra que porta.

Sin embargo los creadores de estos cascos aun no los terminan puesto que se necesitan muchos recursos.

Capítulo 3 Desarrollo

3.1 Planteamiento del problema

Con base en las estadísticas de accidentes fatales en motocicletas surge la necesidad de proponer un dispositivo que permita incrementar la seguridad al momento de conducirlas.

Si bien, el uso del casco reduce considerablemente las posibilidades de sufrir una lesión fatal no existe la suficiente educación vial que permita un manejo seguro por las diferentes vías terrestres de comunicación. En este sentido se busca que el dispositivo generado brinde al motociclista información y alertas relacionadas con posibles colisiones y que aquel, dada la versatilidad de una motocicleta, pueda tomar las acciones pertinentes para evitar un posible accidente.

Se decide que el dispositivo a desarrollar deberá mostrar, en el visor de un casco de motocicleta, lo que ocurre hacia la parte trasera del motociclista. De esta manera se busca disminuir las pequeñas distracciones al momento de utilizar los espejos laterales.

En esta primera etapa del proyecto, se comprobará si es posible visualizar dos imágenes, provenientes de la parte frontal y posterior de la motocicleta, en forma simultánea (realidad superpuesta) en el visor de un casco y sin que estas causen algún problema en la visión y conducción del motociclista.

Para lograr este objetivo se busca una solución de sistemas ópticos partiendo del problema que colocando un objeto a cierta distancia del ojo (<10 cm) esta no se enfoca.

3.1.1 Requerimientos

Para la realización de este proyecto los requerimientos son:

- Peso, dimensiones y materiales adecuados.
- Sin interferencias con la visibilidad hacia el exterior.
- Permita visualizar una imagen tanto con luz de día como en condiciones de poca luz.
- Alimentación de $12V_{DC}$.
- Que la imagen sea visible en el visor del propio casco.

3.1.2 Visor

Una de las mayores restricciones en este proyecto es poder ver en el visor del casco, una imagen o video de lo que ocurre en la parte posterior del mismo. La distancia promedio entre el ojo y el visor es de 10 cm. El enfoque del ojo se lleva a cabo debido a que la lente del cristalino se aplana o se redondea, a este proceso se le llama acomodación. En un ojo normal no es necesaria la acomodación para ver objetos distantes, pues se enfocan en la retina cuando la lente esta aplanada. Para ver objetos más cercanos el músculo ciliar se contrae y por relajación del ligamento suspensorio, la lente se redondea de manera progresiva.

Un niño puede ver con claridad a una distancia tan corta como 6 cm. Al aumentar la edad, las lentes se endurecen y la visión disminuye hasta unos 15 cm a los 30 años y hasta unos 40 cm a los 50 años.

Con el paso de los años, la mayoría de los seres humanos pierden la capacidad de acomodar sus ojos a distancias cortas. Esta condición se llama presbicia y puede ser corregida utilizando lentes convergentes.

3.1.3 Ideas iniciales

En una primera iteración, se emplean dos diferentes pantallas colocadas a 10 cm del ojo. El propósito es que ambas muestren la imagen proveniente de una cámara ubicada en la parte posterior del casco. Una de las pantallas tiene la característica de que puede desplazarse hacia la derecha para dejar sin interferencia alguna al visor. En ambos casos, el usuario no logra enfocar correctamente la imagen (figura 40).

Figura 40: Primera iteración.

3.1.4 Propuestas para el diseño de un visor

Como consecuencia de la dificultad para enfocar una imagen a 10 cm de distancia, se decide emplear una lente convergente. Para tal efecto se emplea un armazón, figura 3.2, en donde se coloca una sola lente. Con este dispositivo se logra ver una imagen a una distancia entre 7 y 9 cm.

Desafortunadamente, una consecuencia de este dispositivo es que ocasiona jaquecas al usuario. Esto como consecuencia de que los ojos enfocan distintas imágenes a diferentes distancias. Por esta razón se decide realizar una tercera iteración (Fig. 41).

Figura 41: Armazón con lupa.

3.1.5 Divisor de luz (superficie reflejante semitransparente)

Esta superficie tiene las propiedades de reflexión y transparencia, gracias a las cuales se pueden ver dos imágenes al mismo tiempo, la que esta atrás de la superficie, y la que se transparenta.

Después de probar con diferentes materiales al alcance, se opta por una superficie previamente revestida por una fina capa de material semiconductor que incrementa la reflexibilidad de la superficie. Debido a que no se está diseñado como divisor de luz es manejada con el nombre de superficie reflejante semitransparente.

Con la finalidad de evitar jaquecas, se decide modificar el diseño previo. De esta manera, se coloca la lente convergente en forma perpendicular al ojo y se aprovecha la capacidad de reflexión y transparencia de la SRS. En esta siguiente iteración, se emplea una superficie lo suficientemente grande para cubrir el campo de visión de ambos ojos, como se observa en la figura 42.

3.1.6 Diseño del visor

Figura 42: Visor para ambos ojos.

En este punto y con las iteraciones anteriores, los sujetos de pruebas confirman que sí es posible ver una imagen a una corta distancia (menor a 10 cm) y que puede realizar esta actividad sin que sufra malestar alguno. Por estas dos razones, el proyecto sí es viable.

Para reducir el espacio y facilitar su instalación en el casco se coloca un espejo a 90° de la SRS, de esta forma se reduce el espacio que ocupa dentro del casco al momento de colocar la pantalla.

Esta versión en lugar de usar una lupa convencional como lente convergente, usa una lupa plana llamada lente de Fresnell que ocupa menor espacio.

Para integrar el visor en el casco se crea un chasis de acero inoxidable donde se montan los componentes, véase figura 43.

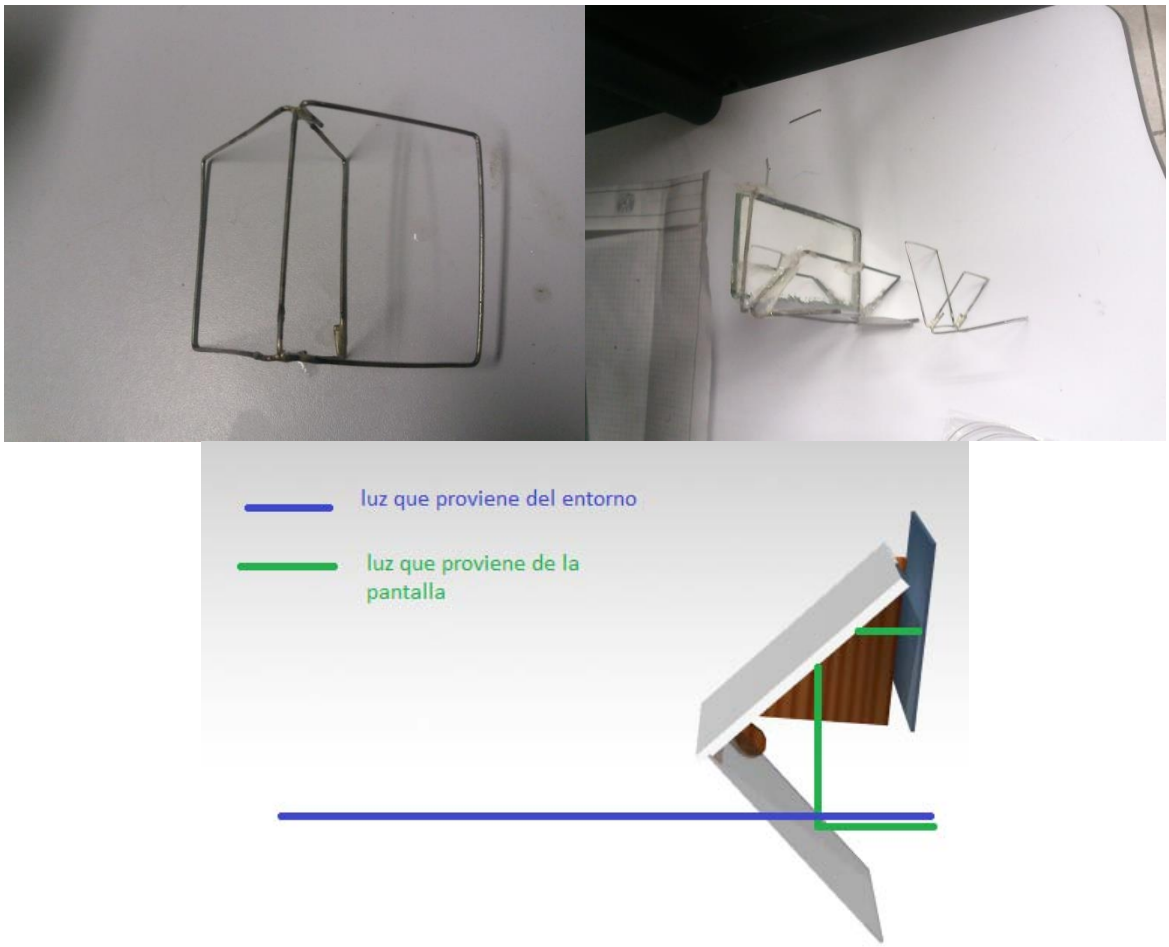


Figura 43: Visor para un ojo y diseño en CAD.

Un nuevo chasis de MDF permite que la superficie reflejante semitransparente esté sujeta de un borde y no tenga un marco que obstruya la visión.

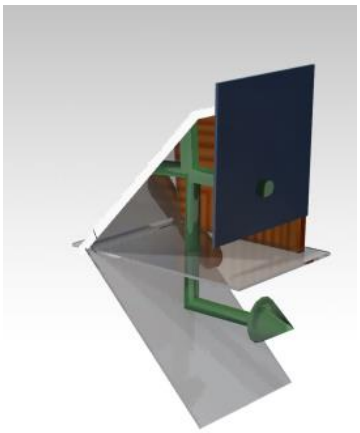


Figura 44: Funcionamiento ideal del Visor para un ojo.

Para realizar pruebas se construye un visor más grande para colocarse afuera del casco (véase figura 45). En esta versión la pantalla puede ser enfocada sin ayuda de una lente.

Figura 45: Visor externo sin lentes.

3.1.7 Calculo de la lente

Tras una revisión de las características de la lente se encuentra que las dimensiones del visor no coinciden con el foco de la lente, por lo que el visor necesita ser más largo.

Para diseñar la lente se parte de la ecuación de las lentes delgadas (ecuación 2):

$$\frac{1}{\mathrm{f}} = \frac{1}{\mathrm{S}_2} - \frac{1}{\mathrm{S}_1} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots ec.2$$

y resolviendo para S_2

$$\mathrm{S}_2 = \frac{\mathrm{FS}_1}{\mathrm{F} + \mathrm{S}_1} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots ec.3$$

Donde S_1 es la distancia de la pantalla a la lente, S_2 es la distancia de la lente a la imagen virtual de la pantalla y F es el foco de la lente.

La lente de la Figura 3.5, tiene las medidas de S_1 =-5cm, F= 30cm. Lo que da una distancia de la lente a la imagen de S_2=-6cm. Con esta longitud junto con S_1 se tiene una distancia de 11 cm por lo que a esta distancia no se puede observar la imagen.

Incrementando S_1 a -15 cm, se logra que la lente pueda ser colocada en el casco. Con la lente anterior (F=30cm) la nueva distancia es S_2 de 30cm, pero ésta todavía se encuentra muy cerca del ojo.

Incrementando la potencia de la lupa, cambiando el foco a F=20cm se tiene una distancia S_2 de 60cm. Esta distancia es la más adecuada ya que se requiere menor esfuerzo para enfocar la imagen obtenida después de haber enfocado objetos distantes.

Para obtener las dioptrías a partir del foco, se usa la siguiente ecuación:

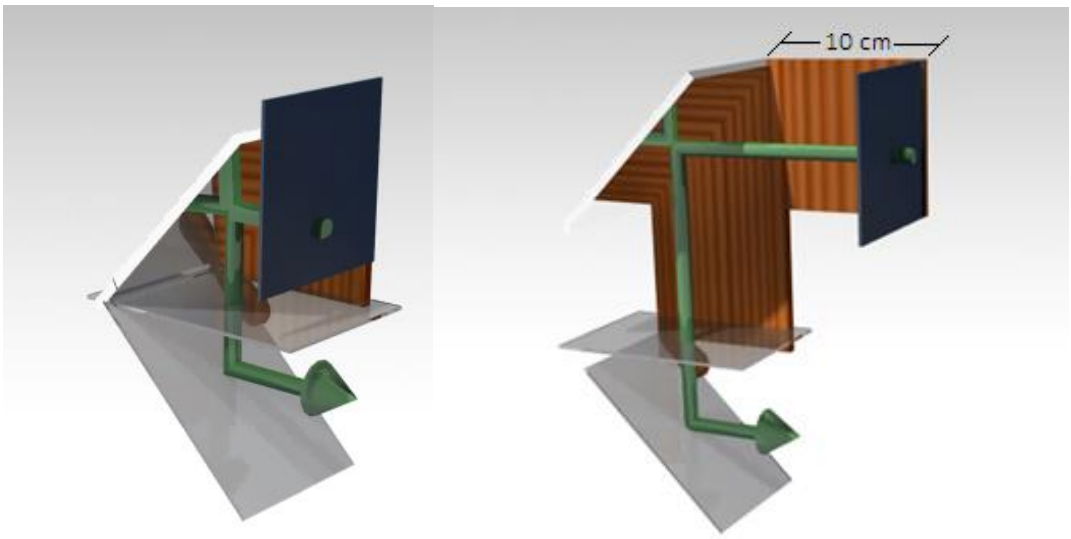
$$\mathrm{D} = \frac{1}{\mathrm{F}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots ec.4$$

Con esta ecuación se realizan algunos cálculos (tabla 3.1) para determinar las características de la lente que mejor cumpla con las características apropiadas para el proyecto.

	Foco[cm]	Distancia de la pantalla a la lente[cm](s_1)	Distancia de la lente a la imagen virtual generada[cm](s_2)	Dioptrías
Visor inicial	30	-5	-6	3.33
	5	-15	7.5	20
	10	-15	30	10
	15	-15	#¡DIV/0!	6.66
Visor corregido	20	-15	-60	5
	25	-15	-37.5	4
	30	-15	-30	3.33
	35	-15	-26.25	2.85

Tabla 3.1. Relación de medidas del visor con el foco de la lente y la visibilidad de la pantalla.

Debido a la forma del visor, se agregan 10 centímetros de la forma como se indica en la Figura 46.



a). b).

Figura 46: a) versión sin corregir. b) versión corregida.

3.1.8 Pantalla

Para mostrar la imagen de la parte posterior del usuario se emplea una pantalla LCD de 4 pulgadas. Las principales características de esta pantalla son: contiene video compuesto (RCA), se alimenta con 12 [V] y consume 120 [mA], tiene capacidad para ajustar el brillo, y se puede invertir el video en modo espejo.

3.1.9 Lámpara

La superficie semitransparente reflejante pierde visibilidad si la cantidad de luz que se refleja es mucho menor que la cantidad de luz que la atraviesa (luz ambiental), este problema se hace presente en el día cuando el cielo está despejado. Para corregir este problema se tienen dos alternativas:

- Incrementar la cantidad de luz que sale de la pantalla.
- Disminuir la cantidad de luz ambiental.

Así, se decide incrementar la cantidad de luz de la pantalla, aplicando una fuente de luz led.

El encapsulado de leds perteneciente al modelo 5050 el cual tiene 3 leds, se elige por la cantidad de lúmenes, es de montaje superficial, es económico y ocupa poco espacio. Véase Figura 47

Figura 47: Conjunto de Leds Ultra brillantes.

La hoja de datos muestra que demanda una corriente de 60 [mA] y funciona con 4 $[V]_{DC}$.

La fuente de luz es una tablilla fenólica perforada, a la cual se le montan 24 módulos de leds 5050 distribuidos en 6 hileras por 4 columnas. Dando un total de 72 leds. Las 6 hileras se agrupan en 3 grupos, cada grupo tiene 8 leds 5050 conectados en paralelo, estos 3 grupos se conectan en serie para ser alimentados por la batería de 12 [V]. En el circuito se usa una resistencia de 2 ohm. El multímetro comprobó que la fuente de luz consume 320 [mA], de forma que cada módulo 5050 consume 40 [mA].

Todos estas partes se montan como se muestra en la Figura 48.

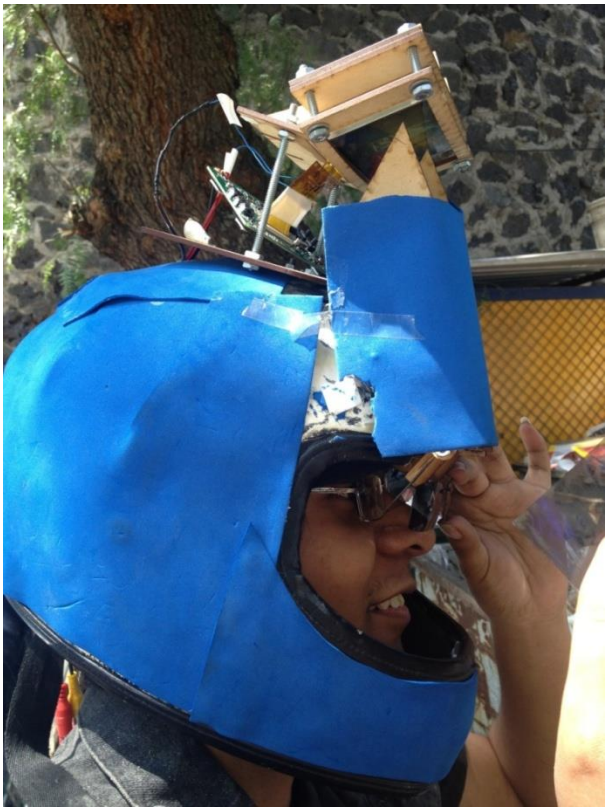


Figura 48: Forma final del visor, las medidas pueden consultarse ver en el apéndice 1.

3.2 Propuesta con PIC32

Una de las primeras ideas para realizar el proyecto fue usar una cámara para transmitir los datos a un microprocesador y de este a la pantalla de una tarjeta de desarrollo. Para ello se consideran los siguientes componentes:

La cámara TCM8240. Esta cámara puede capturar hasta 15 cuadros por segundo, con una resolución desde 160x120 (QQVGA) hasta 1280x1080. La cámara ocupa dos entradas de alimentación del voltaje una a 2.8V y otra a 1.6V y se comunica mediante dos puertos, uno es serial I2C y otro es paralelo de 11 bits.

Para usarla se suministra un voltaje de alimentación y después, mediante protocolo I2C se configura la resolución y cuadros por segundo. La salida de la cámara es por un puerto paralelo de 11 pines, de los cuales 8 pines dan un byte de datos, otro pin da la salida de reloj y los últimos dan el reloj de sincronización horizontal y vertical. Cada dos ciclos del reloj la cámara manda 2 bytes y cuando se combina se obtiene el color de un pixel cuya profundidad es de 16 bits. Los pines del reloj de sincronización horizontal y sincronización vertical se activan cada vez que el pixel cambia de línea o termina de mandar una imagen.

La cámara ocupa un oscilador de 20 MHz como reloj principal y a alta resolución puede enviar más de 3 Mb por segundo. Puede comprimir la salida a formato de imagen jpg, reduciendo el tamaño de la imagen reduciendo así la velocidad de transferencia de una foto.

Como microprocesador se emplea el PIC32MX512L debido a que cualquier otro microprocesador accesible no tiene la velocidad necesaria para capturar los datos de la cámara y entregarlos a la pantalla, además de tener la característica que se puede conseguir en una tarjeta de desarrollo que incluye el cristal, los reguladores y las resistencias necesarias dejándolo casi listo para usarse (véase apéndice 2 a 4 para programar y usar el PIC).

La mayor desventaja es que al momento de usar el PIC se requiere de una configuración específica, además de que es recomendable un software de Microchip (versión plus de PIC-C).

La pantalla pertenece a una tarjeta de desarrollo que es controlada por un PIC18FJ.

3.3 Propuesta con Raspberry Pi

Se utiliza una Raspberry Pi (en adelante RPi) debido a que es mejor que el PIC y se le pueden agregar más funciones y dispositivos.

La RPi tiene un listado particular de cámaras que acepta, ya que solo instalando controladores que sean aceptables para la RPi se hice funcionar una cámara web genérica.

A continuación se muestra un listado de las cámaras aceptadas por la RPi.

Marca	Nombre	Número de modelo	Versión verificada de O.S.
Canyon		CNR-WCAM820	Raspbian/wheezy
Creative	Live! CamSocialize HD	VF0610	Raspbian/wheezy
Hércules	Webcam Deluxe		Raspbian/wheezy +Arch
Logitech	QuickCam Messenger	V-UM14	Raspbian/wheezy +Arch
Microsoft	LiveCam HD-3000	HD-3000	Raspbian/wheezy
Microsoft	LifeCam	VX-1000	Raspbian/wheezy

Tabla 3.2. Cuadro de cámaras aceptadas.

De acuerdo al fabricante de la *Raspberry Pi*, se aconseja usar un sistema operativo basado en *Debian* (sistema operativo que está diseñado para una computadora x86), *Raspbian* está diseñado para computadores con procesadores tipo ARM.

Los programas utilizados están hechos en *Python*, un lenguaje de programación de alto nivel y con esto, se hace que una cámara web genérica funcione en *Raspbian*.

En la modificación del archivo del gestor de arranque llamado *init.d* se modifica una línea la cual hace que el sistema inicie sesión automáticamente pero en modo texto.

También en la misma dirección del archivo *init.d* se le coloca un *script* el cual hace que se ejecute el programa hecho en *Python* para que inicie la cámara automáticamente sin intervención del usuario.

Para ver detalles de los cambios aplicados véase los apéndices 6, 7 y 8.

Para hacer funcionar la cámara web genérica se instalan controladores de video que se toman de programas que generalmente funcionan en distribuciones basadas en Debian 6, en este caso Ubuntu 12.04 y con ello se busca el similar en la *RPiStore* o en las páginas de dichos programas como *UvcCapture.*

Se instala en *Raspbian UvcCapture,* basándose en el manual e instalación por terminal en Ubuntu. *UvcCapture* sirve para tomar fotos y video, además de reconocer ciertos controladores que generalmente son desconocidos por ser de marcas no reconocidas (genéricos). *UvcCapture* funciona mediante terminal.

Se inician las pruebas con un lenguaje de programación instalado de fábrica en *Raspbian*, nos referimos a *Python* 2.7.

Con este lenguaje de programación se hace un programa que capture video (*streaming*) en Ubuntu, para después adaptarlo a *Raspbian* mediante un módulo llamado *VideoCapture* y *SimpleCV*, este último simplifica el uso de *VideoCapture*, de forma que el programa solo tiene que compilarse para que la cámara funcione.

Después de desarrollar el programa para el video, se logra hacer por medio de una modificación al registro de inicio, que el programa se inicie automáticamente y el usuario solo tenga que ocuparse de encender y apagar como si fuese una cámara normal; claro que a la RPi se le pueden agregar más aplicaciones como se muestra en capítulos más adelante.

3.4 Propuesta con Mini PC

La Mini PC es una maquina basada en un núcleo ARM rk3066 el cual contiene dos procesadores, 1 Gb de RAM y 8 Gb de memoria. Se utiliza para sustituir a la RPi. Ya que es más rápida debido a su aumentada memoria RAM, además es mucho más fácil de usar porque tiene como sistema operativo Android 4.0.1. La Mini PC procesa imágenes más rápido que la RPi.

Para utilizarla se realiza el mismo procedimiento que en la RPi, es decir, se crea una librería que permite modificar a un archivo interno al cual se le añade una indicación específica para inicializar la cámara, dándole prioridad a esta al encender la Mini PC y así el usuario no tenga que hacerlo manualmente.

Como la Mini PC tiene incluida una cámara, al programa le es más fácil encontrarla y darle las prioridades necesarias en el sistema.

Capítulo 4 Resultados

4.1 Resultados de la propuesta con PIC

Para hacer pruebas con la cámara se decide separar las funciones que constan en:

- Inicializar la cámara:

Para inicializar la cámara se usa un PIC18FJ, el cual ya está integrado junto a una pantalla en una tarjeta de desarrollo (véase Figura 49) y después de encender la cámara como se indica en la hoja de datos se obtiene la señal de salida que también se indica en la misma.

Figura 49: Tarjeta de desarrollo con PIC18FJ.

Capturar datos de la cámara:

Se hace un sencillo programa para capturar los datos de la cámara (véase apéndices 3 y 4), sin embargo al probar la cámara, esta no logró encender. Se cree que apareció una descarga electrostática o una falla en la fuente de alimentación ocasionando la descompostura de la cámara.

4.2 Resultados de la propuesta con RPi

El sistema tiene un buen funcionamiento. Se hacen pruebas con un usuario manejando una bicicleta y se desarrolla adicionalmente un sistema de alimentación portable, se adapta la cámara a la misma bicicleta.

Se presentaron problemas de compatibilidad de la cámara, como consecuencia de que la RPi tiene problemas para procesar una resolución de video mayor a 160x120 pixeles.

En este punto, varias personas prueban el Sistema de Realidad Superpuesta y comentan que se pueden enfocar la imagen secundaria perfectamente, sin causar malestar aparente o dolor de cabeza.

A continuación se muestra en la Figura 50 como se logra de primera instancia llegar a iniciar el sistema operativo en modo gráfico.

Figura 50: Iniciando el S.O. Raspbian.

En la Figura 51, se muestra la ejecución de la cámara, que también está implícita la modificación de inicios automáticos de programas y del Sistema operativo

Figura 51: Iniciando la cámara con Python.

4.3 Resultados de la propuesta con Mini PC

Cuando se usa la Mini Pc se hacen varias adaptaciones para la comunicación hacia la pantalla y se hace un circuito de alimentación, dicho circuito no solo alimentara la computadora sino que también a la pantalla y a los leds detrás de esta última.

Se proponen hacer dos pruebas a diferentes horas del día para probar la funcionalidad. La primera prueba se realiza en la noche. El usuario comenta que existe una falta de visibilidad debido al destello luminoso creado cuando la cámara apuntaba hacia alguna fuente luminosa, como los faros de los coches o las luminarias. Pese a este inconveniente, con este sistema de visión de realidad superpuesta se logra distinguir si algún vehículo proviene de la parte trasera y simultáneamente se logra ver hacia el frente, logrando así enfocar dos imágenes distintas.

La falta de visibilidad en este caso se debe a que el diseño de la cámara es para usarse en video conferencias. Una cámara diseñada para exteriores puede tomar una imagen mejor.

La segunda prueba se realiza en el día. El usuario comenta que la imagen es un tanto difusa a causa de la luz ambiental. En caso de emplear una lente obscura es fácilmente observable.

El efecto de la S.R.S depende de la cantidad de luz que entra tanto del exterior del casco como de la pantalla. El reflector de 72 leds carece de la intensidad de luz suficiente para contrarrestar la luz ambiental producida por el sol; efecto similar al que ocurre con una pantalla de computadora portátil, pues en ciertas condiciones de luz natural es difícil observar las imágenes dentro de la misma.

4.4 Comparación de resultados

	Ventajas	Desventajas
PIC	El mercadeo está permitido por el fabricante del PIC, así como del software de programación empleado.	Requiere de conocimientos de programación específica del dispositivo, la cámara es mucho más delicada y necesita el diseño y desarrollo de un circuito eléctrico. La falta de capacidad de procesamiento para operaciones complejas de video.
Raspberry Pi	Es más barata. Es más fácil de programar. Tiene capacidad de procesamiento para operaciones complejas de video. Sistema operativo basado en Debian 6 permite cambios en las librerías y registros del sistema. Salida de video HDMI y RCA.	Velocidad de procesamiento limitada. Necesita ventilador para overclock. Necesita conocimientos previos de Debían. La cámara debe ser de un modelo específico para ser compatible. No incluye sistema operativo.
Mini-PC	Cámara incluida. Salida de video HDMI y AV. Mayor velocidad de procesamiento. Incluye sistema operativo Android listo para usarse. Tiene una carcasa para proteger el circuito eléctrico. Es más fácil agregar programas de terceros. Existen varios modelos para diferentes necesidades.	Requiere de conocimientos de varios lenguajes de programación de alto nivel. Cámara incluida es de baja calidad. Por políticas de versión de Sistema operativo, es de difícil acceso a los registros del sistema. Tiene restricciones para alguna modificación al sistema operativo.

Tabla 4.1. Cuadro comparativo de resultados.

Con base en la Tabla 4.1 la mejor opción es la Mini PC por que no se tienen tantas limitantes de hardware, aunque si de software, pero se pueden modificar ciertos archivos sin alterar las políticas del sistema.

Una idea sería utilizar una cámara IP y que estuviese conectada inalámbricamente a la Mini PC, para que esta última procese el video.

Indica mucho costo pero el funcionamiento podría ser mejor.

Capítulo 5 Conclusiones y Trabajo a Futuro

5.1 Conclusiones

En nuestras clases de óptica, se vio que con una lente convexa se corrige la hipermetropía, una persona con este padecimiento no puede ver los objetos cercanos. Así, se intuye que una persona con visión normal y lentes convergentes podría ver más de cerca, lo que se comprobó usando una lupa común (que también es una lente convergente). De esta manera se resolvió uno de los problemas de ver la pantalla adentro del casco. Sin embargo, otro problema es que una persona tiene que realizar un gran esfuerzo para ver la pantalla y no se llegaba al objetivo de superponer imágenes.

Otra de las formas en que se intenta lograr el objetivo es usar un proyector que muestre la imagen en el visor, pero aquel es muy pesado y voluminoso. Se buscaron por internet algunos proyectores de menor tamaño pero estaban afuera del presupuesto.

Ciertas pantallas hechas con tecnología OLED son transparentes y se puede ver a través de ellas al mismo tiempo que presentan su información. Sin embargo al montarse en el casco tendrían el mismo problema de ocasionar dolor de cabeza.

Así se consigue un objeto que tuviera las propiedades ópticas de reflexión y transparencia. Esta superficie hace la función similar a un divisor de luz, pero en lugar de dividir, superpone imágenes, la superficie se pone a cierto ángulo para que refleje la imagen proveniente de la pantalla y que al mismo tiempo se transparente la imagen de enfrente. Con esta superposición de imágenes se logra el objetivo propuesto y se llega al término de realidad superpuesta.

Hubo una serie de iteraciones en la forma del visor para mejorar la visión, así como la calidad de los materiales de los que está hecho, por lo que se concluye que el último visor es la mejor forma de obtener un modelo funcional.

Cuando se empezó el proyecto se tenían muchas ideas de como poder tener un prototipo funcional, pero resultaban muy costosas por lo cual se idearon otras propuestas de solución. En ese momento se decide explorar otras propuestas económicas y más sencillas de realizar; por lo que se decide emplear un PIC, sin embargo este carece de la capacidad de procesamiento suficiente para las necesidades del proyecto (secuencia imágenes de 20

a 25 cuadros por segundo) en el entendido que el ojo humano ve una secuencia de imágenes a más de 30 fps, esto para que se tomara la secuencia como un video común y que no se distrajera el usuario en tener que procesar las imágenes y así diferenciar la imagen trasera de la delantera.

El PIC funcionó parcialmente debido a sus restricciones con el video. Pese a ello se considera que por las ventajas de comercialización es la forma más económica.

Otra alternativa es el uso de las microcomputadoras de núcleo tipo ARM. En este caso se propone una Raspberry Pi, y el fabricante recomienda para su funcionamiento cual sistema operativo (S.O.) usar y cuales funciones puede realizar, así como algunas recomendaciones de hardware que son una pantalla y una cámara. En este sentido, son periféricos que se pueden comprar directamente con el mismo fabricante, sin embargo esto aumentaba el tiempo del desarrollo y el presupuesto. En este punto, la compra de estos productos hubiese simplificado muchas tareas.

Se realizaron muchos experimentos con la RPi, lo que llevó a instalar los diferentes S.O. para probar su uso y sus funciones activas dentro de la RPi. Se coincide que *Raspbian* es la mejor opción debido a su similitud con Ubuntu, el cual es un sistema operativo del que tiene mayor experiencia de uso. De esta manera se configuran y modifican ciertos archivos del sistema, así como su instalación de librerías y paquetes, para que la cama pueda iniciar automáticamente.

Al final fue de gran uso y después de cierto tiempo fue más fácil usar la Raspberry Pi con S.O. *Raspbian*.

Sin embargo se necesita que la minicomputadora tenga mejor procesamiento de video. En esta iteración es sencillo verificar que el principio de realidad superpuesta es válido y es viable. Al igual que el PIC, la RPi carece de procesamiento de video, pero tiene la desventaja de que se requiere comprar los componentes de la misma marca además hasta este momento la RPi no tiene una bahía de expansión para la memoria RAM y depende de una memoria SD para instalar el S.O.

La forma en que funciona el sistema operativo desde una memoria externa es similar a los S.O. de GNU/LINUX que tienen el modo LIVE, donde solo requieren de una memoria RAM para que allí se ejecuten y no dependa de una unidad interna de almacenamiento, es decir, es bueno este modo, pero trae la desventaja que hay que esperar a que se inicialicen todos los dispositivos para que pasando el BIOS se active SD con el sistema.

Finalmente se utiliza una Mini PC principalmente por sus características extras tales son: la cámara, los módulos de bluetooth y wifi, además de no depender de la salida de video

HDMI únicamente, ya que contiene la de AV (video compuesto) y esto da pauta a que se utilice en una pantalla con entrada RCA, también tiene la ventaja de ya traer un sistema operativo (Android 4.1.0) y además que está en una memoria interna, y se le puede ampliar la capacidad de almacenamiento, desafortunadamente en México no se encuentra tan fácilmente.

Con la Mini PC se hace lo mismo que con la RPi, pero se requieren conocimientos de java y Android. A pesar de que es complicada la modificación de ciertos módulos de sistema, al final se loga hacer que realice funciones similares a las de la RPi tales como: iniciar sesión automáticamente e iniciar del mismo modo la cámara interna (en el caso de la RPi la cámara es a través de USB y un modelo no compatible), dicha mejora hizo que la mini PC diera buenos resultados.

Dicha Mini PC tiene la ventaja de compatibilidad de múltiples programas diseñados para el procesamiento digital de imágenes, como OpenCV, y tenemos la esperanza de que se continúe esta tesis y se tengan mejoras y nuevos diseños.

La tecnología y el conocimiento de la Mini PC hacen posible que nuevos programas sean más fáciles de realizar y con ello deja abierto la posibilidad para mejoras al proyecto.

Consideramos que es un proyecto fácil, nos costó trabajo iniciar y tener conocimiento de ciertas tecnologías y lenguajes de programación pero consideramos que nuestro proyecto (si tuviera más presupuesto) estaría a la par de los cascos que realizan en otros países y con 100 veces más presupuesto que nosotros. El principio funciona, cuando iniciamos no teníamos idea que existieran dichos prototipos, sin embargo en cierto modo notamos que no funcionan y/o solo se quedan en ideas.

A partir de los resultados mostrados en el capítulo anterior concluimos que la mejor opción para este proyecto es la propuesta con la mini PC, debido a que a pesar de tener tantas restricciones se encontró la forma de evitarlas sin violar las políticas de Android.

Entonces tenemos como conclusión que el sistema funciona, y que la imagen de la pantalla se ve a corta distancia, no da “dolores de cabeza”, no se requiere tanto esfuerzo y se observan las dos imágenes deseadas (imagen frontal y trasera).

Tiene como desventajas:

- No es seguro usar por el momento debido a que contiene un cristal (lente 5 dioptrías) y en caso de un accidente puede lastimar más al usuario.
- Requiere de baterías voluminosas para alimentar tanto a la pantalla como a la Mini PC.

- La iluminación no es suficiente detrás de la pantalla.
- Se necesita un aparato para disminuir la luz exterior ya que la imagen propuesta no es tan buena.

Consideramos tiene como ventajas:

- Mejorar la visión del usuario ampliándola sin esfuerzo aparente.
- Materiales de fácil acceso.

5.2 Trabajo a futuro

Durante el desarrollo y las pruebas, encontramos que existen muchas cosas que se pueden hacer para mejorar el rendimiento del casco, pero que por razones de tiempo y dinero no se hicieron, pero si se investigaron, las cuales se mencionan a continuación:

5.2.1 Visión artificial usando OpenCV

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta entornos aplicativos de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella indicadas.

Open CV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica.

En nuestro caso se desarrolló un programa que funcionaba por medio de reconocimiento de patrones, para ayudar a detectar objetos como automóviles que estén en nuestra periferia.

Como se muestra en la Figura 52 Este programa toma un color de cierta área y después busca de nuevo el área perteneciente al mismo color, si el área cambia de posición el programa detectara este movimiento.

Figura 52: Seguidor del área verde.

Motempl.

Se encontró en los ejemplos de OpenCV el programa Motempl, este programa detecta direcciones y movimientos (véase Figura 54). En la Figura 53 se aprecia un diagrama de flujo que representa lo que sucede en el programa.

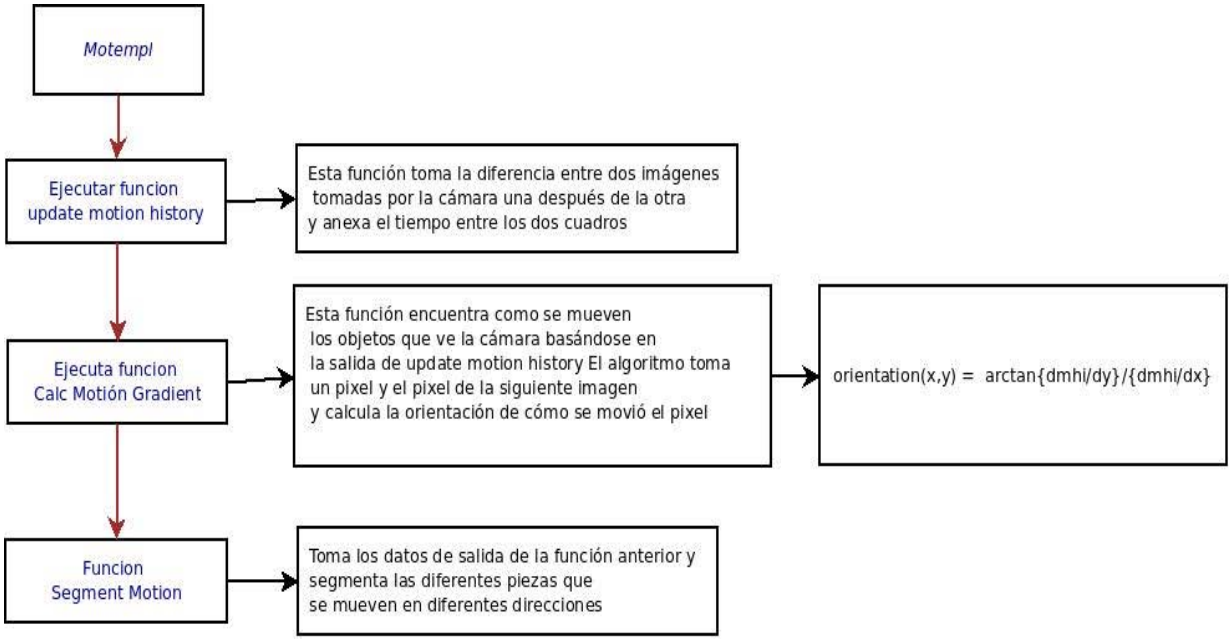


Figura 53: Diagrama explicativo del programa Motempl.

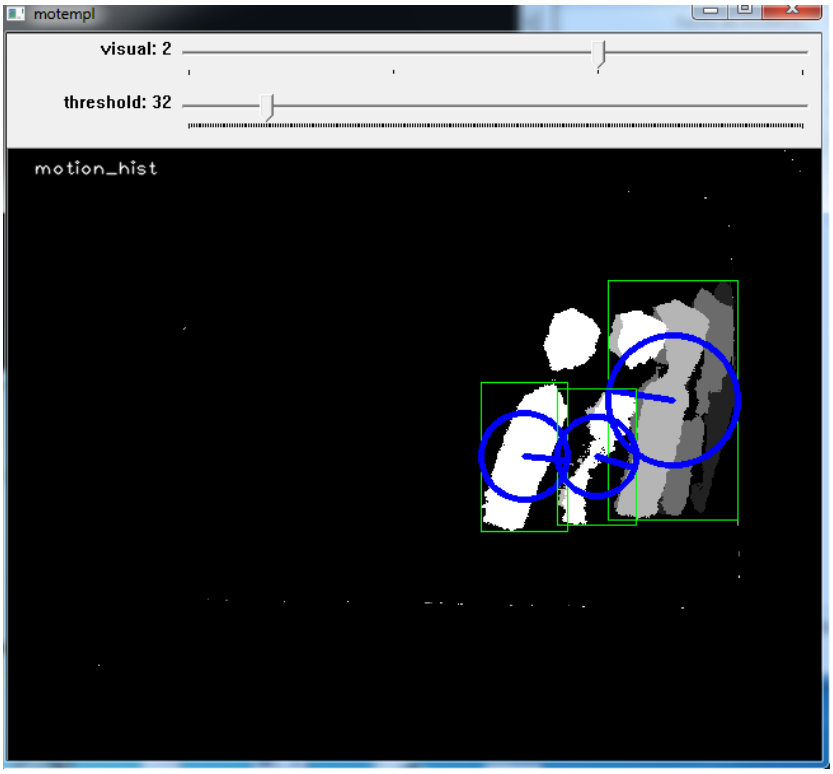


Figura 54: Motempl siguiendo un objeto.

5.2.2 Mini PC y Picuntu

Diversos fabricantes venden modelos diferentes de Mini PC, en nuestro caso escogimos una que tiene salida AV, sin embargo es muy difícil y no está recomendado por el fabricante cambiarle el sistema operativo. Algunas Mini PC están diseñadas para desarrollo y tienen un segundo compartimiento MicroSD para poder cambiar el sistema operativo, incluso existen algunas que ya se venden con esta distribución de Linux preinstalada.

Dicho sistema nos permite desarrollar sistemas de visión basados en OpenCV más fácilmente.

5.2.3 GPS y comunicaciones

Con un punto de acceso a internet por medio de una banda ancha la Mini PC tiene servicios de GPS. Con el módulo de GPS la Raspberry Pi también puede tener esta funcionalidad.

5.2.4 Lámpara de alta luminosidad

Aun con la lámpara de Leds la segunda imagen en el casco no tiene suficiente visibilidad en el día, para solucionar este problema se planteó usar una mica obscura, sin embargo esta no es ajustable a la diferente cantidad de luz a lo largo del día, además de que si el visor está en un solo ojo dificulta la visión en general.

Para hacerlo variable se planteó usar 2 micas polarizadoras una fija y otra que gire, cuando los ejes de polarización estén paralelos, la luz ambiental que entra al visor será máxima. Cuando los ejes de polarización estén perpendiculares la luz ambiental que entra al visor será nula por lo tanto la visibilidad de la segunda imagen se mejorará.

En una nueva investigación se encontró que existen ventanas hechas con material electro cromático. Este material se oscurece gradualmente al recibir una tensión eléctrica, y se aclara cuando la tensión eléctrica cambia de polaridad.

5.2.5 Cámara remota

Cuando se probó la Mini PC se notó que la cámara integrada no tiene la claridad esperada en la noche. Para solucionar este problema se planteó tomar una imagen usando un celular y transmitir el video vía bluetooth o wifi hacia la Mini PC.

Al realizar pruebas se comprobó que la transmisión vía bluetooth es muy lenta y de baja resolución.

Para usar wifi se tiene que hacer una VPN para que se puedan comunicar una a la otra y una VNC para que se puedan transmitir datos una a la otra, y esto se llama red interna.

Wifi Direct

La RPi necesita un módulo de wifi con soporte del protocolo de Wifi Direct para que la transmisión de video vaya de la RPi a la Mini PC (esta computadora ya incluye este protocolo de comunicación) de esta forma se le puede agregar una cámara de mayor resolución y de una mejor visión al usuario, también hace modular el diseño del sistema, un módulo la cámara y otro la pantalla. La pantalla funciona con Mini PC, porque el procesamiento de imágenes es mejor, y la mayor consideración es que lo haría de modo inalámbrico.

También se puede colocar un teléfono móvil con SO Android y soporte de Wifi direct y conectarse a la RPi.

5.2.6 Sensores

Se pueden poner sensores en la moto como velocímetro, de combustible, tacómetro, etc. que se comuniquen con la computadora y que los datos aparezcan en la pantalla.

5.2.7 Acelerómetro

En caso de accidente, con un acelerómetro se puede obtener la trayectoria que siguió la cabeza del usuario así como el punto de impacto. Esta información puede ser de gran ayuda para los médicos.

5.2.8 Estabilidad de la cámara

Durante las pruebas se notó que la cámara tiende a vibrar y da una imagen distorsionada. Una de las soluciones consiste es un estabilizador hecho con un brazo de 3 grados de libertad movido por servomotores y controlados por un microcontrolador. Este brazo detecta los desplazamientos bruscos gracias a un acelerómetro y los anula basados en el programa.

Otra solución es hacer un sistema mecánico con resortes, masa y amortiguadores. Este sistema absorberá los desplazamientos bruscos.

5.2.9 Alta resolución

Una cámara de mayor resolución permite ver una imagen más nítida, una idea que se propuso es usar una cámara infrarroja para poder ver de noche.

5.2.10 Seguridad del casco

El visor actual pasa a través del casco comprometiendo mucho la resistencia y la seguridad. Un nuevo visor necesitara ser más pequeño, tener una pantalla más pequeña y preferiblemente estar montado afuera de la superficie dura del casco, además de tener la superficie reflejante semitransparente en la visera del casco.

Referencias

BIBLIOGRAFIA

[8] DI JASIO, LUCIO. PROGRAMMING 32-BIT MICROCONTROLLERS IN C: EXPLORING THE PIC32. U.S.A. ELSEVIER SCIENCE & TECHNOLOGY (2008-05).

[9] EL GRAN LIBRO ANDROID DE TOMÁS GIRONES, JESÚS. EDITORIAL: MARCOMBO, S.A.

[10] ÓPTICA, HECHT EUGENE, TERCERA EDICIÓN EDITORIAL: PEARSON, ADDISON-WESLEY, AÑO 2003.

[11] FUNDAMENTOS DE ÓPTICA, F.A. JENKINS Y H.E.WHITE EDITORIAL: AGUILAR 1964.

[12] MICROCONTROLADOR PIC18F84. PALACIOS, ENRIQUE. MÉXICO. ALFAOMEGA 2004

[13] PROGRAMACION SHELL EN UNIX/LINUX SH (BOURNE), KSH, BASH, CHRISTINE DEFFAIX REMY, ENI, 2010.

[14] EL LIBRO DEL ADMINISTRADOR DE DEBIAN WHEEZY, RAPHAEL HERTZOG.

[15]BASH SHELL SCRIPTING GUIDE FOR BEGINNERS.

[16]THE COMPLETE ANDROID GUIDE, KEVIN PURDY.

[17] JAVA PARA DESARROLLO ANDROID, JEFF FRIESEN, ANAYA MULTIMEDIA, 2011.

MESOGRAFIA

[1] www.oni.ecuelas.edu.ar

[2] www.who.int/mediacentre/factsheets/fs358/es/

[3] demotosonline.com/category/cascos-de-moto/page/3/

[4] www.motorpasionmoto.com/seguridad/breve-historia-del-casco

[6] Manual para Pic 32:

http://ww1.microchip.com/downloads/en/DeviceDoc/61127D.pdf

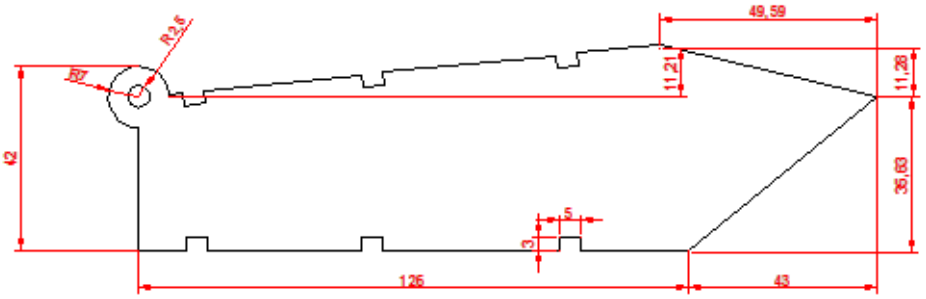
[7]https://www.sparkfun.com/datasheets/Sensors/Imaging/TCM8240MD_APPNOTE041214.pdf

http://www.cnx-software.com/2012/11/04/rockchip-rk3066-rk30xx-processor-documentation-source-code-and-tools/
Figura 31: Teléfono con aplicación de realidad aumentada.
http://informatic2you.wordpress.com/2013/03/18/fases-realida-aumentada/
Figura 32: Ojo con lente de Realidad Aumentada.
http://nadanoslibradeescorpio.blogspot.mx/2012/02/vizio-cinemawide-58-lentes-de-contacto.html
Figura 33: Piloto con casco BAE.
http://www.bbc.co.uk/mundo/noticias/2012/09/120906_tecnologia_cascos_realidad_virtual.shtml
Figura 34: Casco del F-35.
http://thebrigade.thechive.com/2014/03/23/go-future-with-the-f-35-helmet-22-hq-photos/
Figura 35: Casco HEaDS-UP.
http://www.todotechie.com/heads-up-el-casco-inteligente-del-ejercito-de-usa/
Figura 36: Casco JHMCS.
http://razonyfuerza.mforos.com/549912/11318250
Figura 37: Visión ideal del casco Reevu MSX1.
Figura 38: Visión real del casco Reevu MSX1.
www.reevu.com/
Figura 39: Casco ideal LiveMap.
http://vroom.tudiscovery.com/livemap-casco-para-una-nueva-era-de-motociclistas/
Figura 49: Tarjeta de desarrollo con PIC18FJ.
mikroe.com
Figura 60: Indicación 1-1.
Figura 61: Indicación 1-2.
Figura 62: Indicación 2.
Figura 63: Indicación 3-1.
Figura 64: Indicación 3-2.
Figura 65: Indicación 5-1.
Figura 66: Indicación 5-2.
Figura 67: Indicación 5-3.
Figura 68: Indicación 5-4.
Figura 69: Indicación 5-5.
http://www.schmalzhaus.com/UBW32/doc/How-To%20Compile%20Projects%20for%20the%20UBW32.pdf

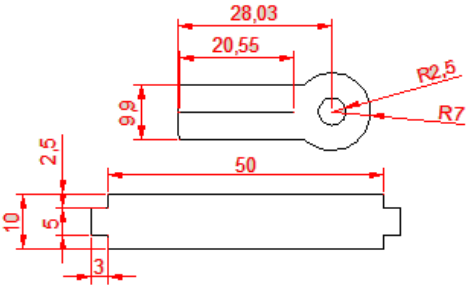
Apéndices

Apéndice 1 Detalles de la construcción del visor y soporte de la pantalla

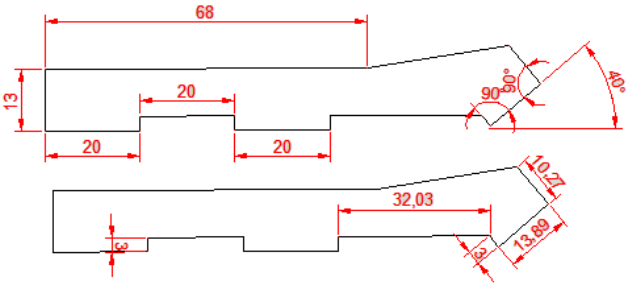
Para acoplar los elementos necesarios se creó una carcasa que soporta y fija los elementos al casco, esta carcasa está hecha de MDF (*médium density fibrocell*) de 3mm cortada en laser según la Figura 55.



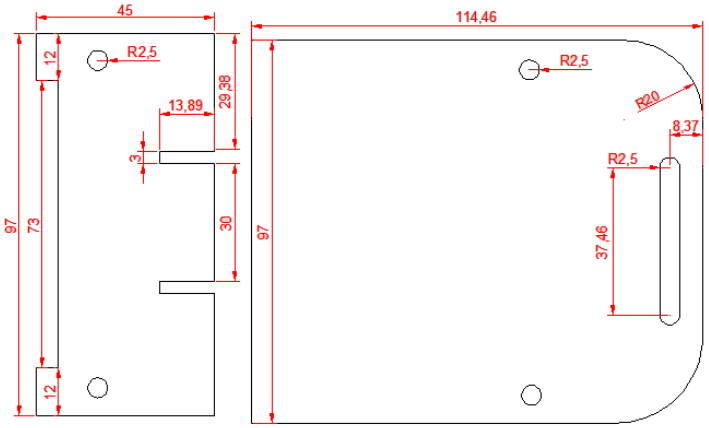
Parte lateral del visor.



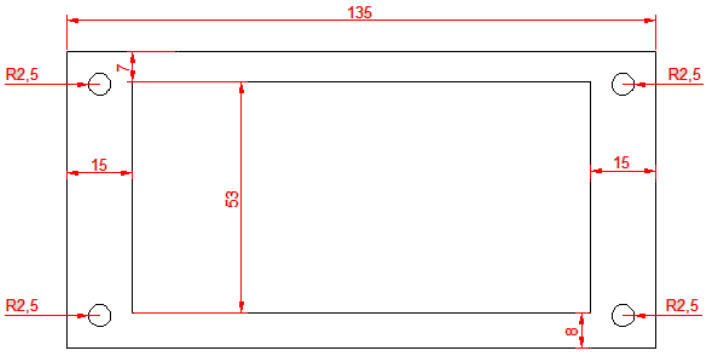
Travesaño y sujetador de la superficie refleiante.



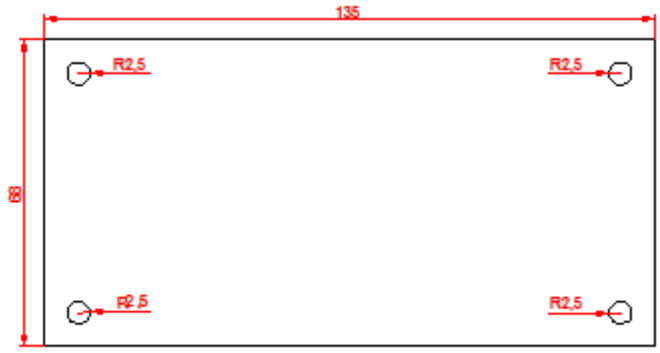
Soporte inclinado para pantalla.



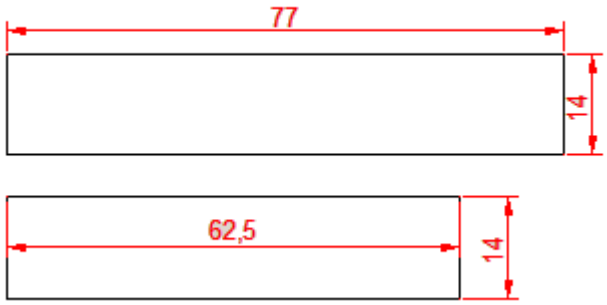
Soporte inferior de pantalla y base para acoplar al casco.



Marco frontal de la pantalla.



Marco posterior de la pantalla.



Separadores.

Figura 55: Medidas de los elementos del visor.

Apéndice 2 Como leer los datos de la cámara

En la hoja de datos vienen diagramas similares a la Figura 56.

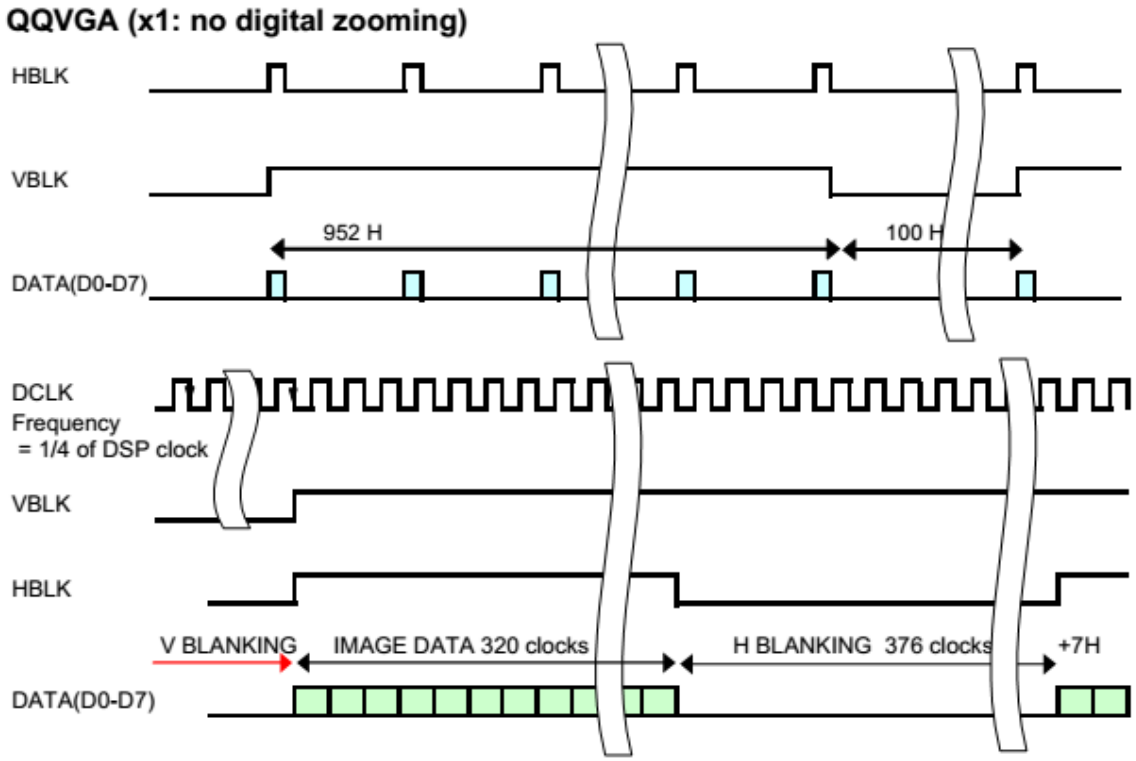


Figura 56: Diagrama de Lectura de datos de la cámara.

El tamaño de imagen para esta resolución (QQVGA) es de 160 pixeles de ancho por 120 pixeles de alto.

Por cada ciclo del pin DCLK (data clock (reloj de datos)) los pines d0-d7 envían un bit cada uno. De esta forma se genera un byte.

La cámara envía 2 bytes consecutivos para formar el color de un pixel de 16 bits de profundidad.

Por lo tanto en 320 ciclos del reloj de datos la cámara envía 160 colores de pixel que corresponden a una línea horizontal de la imagen que se ha capturado.

En este periodo de 320 ciclos del reloj de datos los pines HBLK (horizontal blank) y VBLK (vertical blank) permanecen en estado alto.

Después pasan 376 ciclos del reloj de datos (ciclos del pin DCLK) durante los cuales los pines de data d0 a d7 y el pin HBLK estarán apagados.

Estas dos últimas partes se juntan y forman un ciclo de 696 ciclos del reloj de datos para formar una unidad de tiempo llamada 1H.

A continuación seguirán 7H ciclos durante los cuales os pines de data d0 a d7 y el pin HBLK seguirán apagados.

Estas 3 partes se repetirán 119 veces y terminaran con un ciclo de 320 ciclos de reloj de datos durante los cuales VBLK estará encendido. De esta forma tenemos 120 líneas de 160 pixeles que crearan una imagen de 160x120.

Finalmente pasara una parte de tiempo igual a 100H durante los cuales tanto VBLK y HBLK estarán apagados. Y se reinicia el ciclo para unos nuevos 320 ciclos del reloj de datos.

Apéndice 3 Como conectar la cámara

La cámara tcm8240 es un dispositivo de montaje superficial, para usarla se requiere soldarla a un circuito impreso.

Se creó uno, basándose en el diagrama de la Figura 57 de la cámara que se incluye en la hoja de datos.

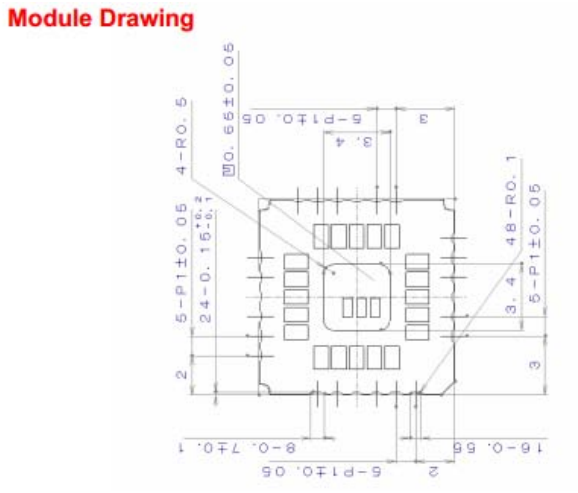


Figura 57: Plano de la cámara.

En el diseño del circuito impreso se tomaron en cuenta las siguientes especificaciones:

- Que se pudiera usar cómodamente con una protoboard.
- Acceso a los pines de datos por un lado del circuito impreso.
- Fuente doble integrada para alimentar al módulo de la cámara.
- Que sea de 1 capa.

Se realizó un circuito impreso mostrado en Figura 58 tomando como referencia la Figura A.3 anterior y usando el software Eagle.

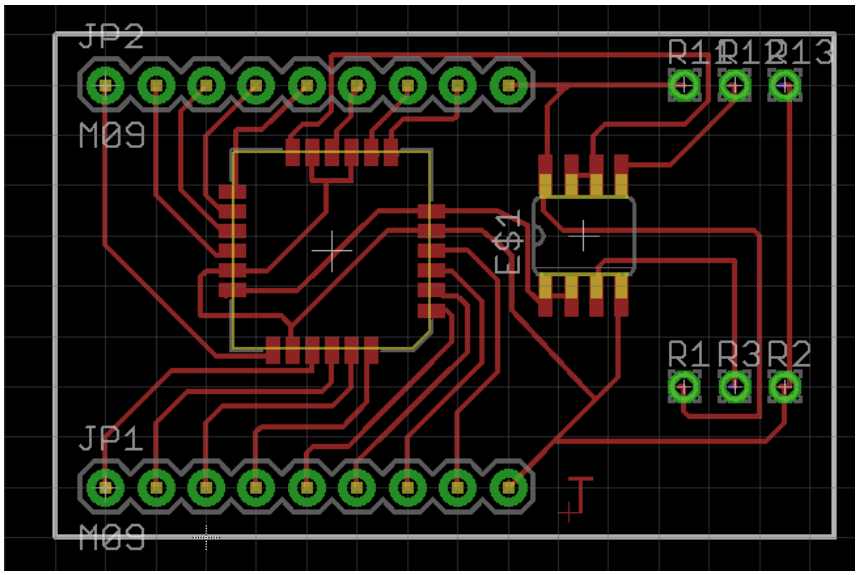


Figura 58: Circuito impreso del Controlador de la cámara.

Debido a método de creación de las pistas del circuito impreso, el diseño se modificó para que tuviera pistas más gruesas dando como resultado final la Figura 59.

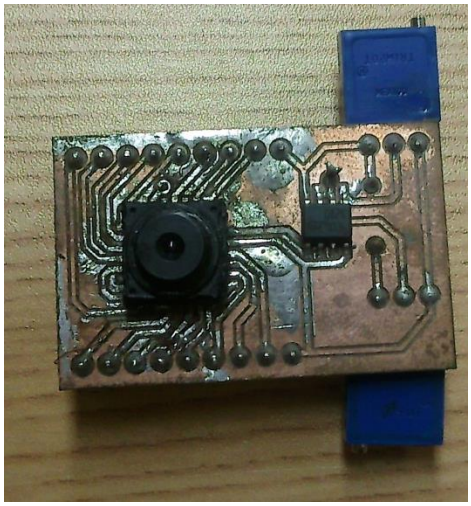


Figura 59: Cámara soldada en la tarjeta de desarrollo.

Fuente doble.

La cámara consume un máximo de 145 mA [datasheet], sin embargo la fuente integrada en el circuito impreso solo ofrecía 20 mA, por lo cual se elige alimentarla externamente.

Uso de la cámara

Para encender la cámara se necesita ingresar comandos vía I2C mediante un micro controlador, los registros que se usaron y su descripción se muestran en la siguiente tabla (véase el documento app note).

Dirección	Valor	Descripción
0x02	0x00	Activar la cámara
0X0B	0X00	Apagar línea blanca que se genera horizontalmente
0X58	0X20	Tiempo de exposición (3/100 s)
0X05	0x00	Ajuste rápido de cuadros por segundo (Valor de ¼ del máximo de cuadros por segundo)
0X1A	0XFF	Byte menos significativo del ajuste de pixeles horizontal (H_COUNT)(valor total=1700)
0X1B	0XB3	Byte menos significativo del ajuste de líneas de pixeles vertical (valor total=1052)
0X11	0X4A	Ajuste de tamaño de imagen (tipo QQVGA 160x128 pixeles)
0X14	0X33	
0X04	0X0D	El formato de imagen es RGB, Encender pines de salida
0X1F	0X0B	Ajuste de pulso interno (963) (véase app note)
0X1E	0XC3	
0X0E	0XAC	Ajuste de tamaño de imagen (tamaño QQVGA)

Tabla A.1 comandos I2C para configurar y encender la cámara.

Pic32

Al principio se usó la tarjeta de desarrollo ubw32 pues tiene el PIC32MX512L que tiene una velocidad suficiente para sacar datos de la cámara y entregarlos a una pantalla.

Al hacer un programa en MikroC, otro lenguaje de programación de micro controlador, hubo el problema de que el bootloader no aceptaba el archivo generado por MikroC.

En este caso, se usa la tarjeta de desarrollo PIC18FJ que ya cuenta con el cableado para hacer la conexión a la cámara mediante I2C. De esta forma rápidamente se logra encender la cámara y cuando se conecta al osciloscopio se observa que la señal de salida era coherente con la que se mostraba en la hoja de datos.

Una vez que se termina de escribir un programa para obtener datos de la cámara usando la tarjeta PIC18 la cámara muestra una falla, marcaba un corto circuito entre las terminales de alimentación y tierra.

Funciones principales del programa para iniciar y leer datos de la cámara:

```
void  configurapuertoscamara(){

  //iniciar puertos
  portb.b6=0;
  latb.b6=0;
  porte.b1=0;
  late.b1=0;
  porte.b2=0;
  late.b2=0;
  porte.b3=0;
  late.b3=0;
  porte.b4=0;
  late.b4=0;
  porte.b5=0;
  late.b5=0;
  porte.b6=0;
  late.b6=0;
  porte.b7=0;
  late.b7=0;
  portg.b1=0;
  latg.b1=0;
  portg.b2=0;
  latg.b2=0;
  portg.b3=0;
  latg.b3=0;
  portg.b4=0;
  latg.b4=0;

  //abrir puertos
  trise.b1=1;         //puerto data  1-7
  trise.b2=1;         //puerto data  1-7
  trise.b3=1;         //puerto data  1-7
  trise.b4=1;         //puerto data  1-7
  trise.b5=1;         //puerto data  1-7
  trise.b6=1;         //puerto data  1-7
  trise.b7=1;         //puerto data  1-7
  trisg.b4=1;        //puerto data  0
  trisg.b2=1;        //vertical blank
  trisb.b6=1;        //horizontal blank
```

```
  trisg.b3=0;         //reset

}
// funciones para I2C
void iic2send(short direccion,short valor){
 i2c2_start();
 i2c2_wr(temp);
 i2c2_wr(direccion);
 i2c2_wr(valor);
 i2c2_stop();
 delay_us(10);
}

void inicializarcam(){
  delay_ms(10);
 // 0x02 0x00  //activar camara
 iic2send(0x02,0x00);
 //0x0B 0x00  // White Line OFF
 iic2send(0x0B,0x00);
 //0x58 0x20  //  tiempo de exposicion
 iic2send(0x58,0x20);
 //0x05 0x00  // Frame Rate
 iic2send(0x05,0x00);
 //0x1A 0xFF  // HCOUNT = 0x3FF = 1023
 iic2send(0x1A,0xFF);
 //0x1B 0xB3  // VCOUNT = 0x21B = 539 // Registro 0x1C esta en default (0xA1)
 iic2send(0x1B,0xB3);
 //0x11 0x4A  // cambia por PICSIZE
 iic2send(0x11,0x4A);
 //0x14 0x33  // cambia por PICSIZE
 iic2send(0x14,0x33);
 //0x04 0x0D  // RGB  352x288  OUT ON
 iic2send(0x04,0x0B);  //qqvga
 //0x1F 0x0B  // SPCOUNT = 0xBC3 = 3011
 iic2send(0x1F,0x0B);
 //0x1E 0xC3  // SPCOUNT[7:0]
 iic2send(0x1E,0xC3);
 //0x0E 0x1E  //
 iic2send(0x0E,0xAC);
}

void camara(){

  tft_fill_screen(0);
```

```
  configurapuertoscamara();



  //inicializar  comunicación I2C
  trisd.b5=0;
  trisd.b6=0;

  i2c2_init(100000);

    //encender pin reset de la cámara

  latg.b3=1;
  delay_ms(100);

  temp=122;     //direccion física es temp que a su vez es global
  inicializarcam();
  x=0;
  y=0;
  while(bandera1){
     tft_dot(x,y,porte);
     x++;
     if(x>=160){
     x=0;
     y++;
     }
     if(y>=120){
     bandera1=0;
     }
  }
}
```

Apéndice 4 Como hacer que mplab funcione con ubw32

1.- Ir a la página de microchip y descargar el instalador para microchip mplab (Fig. 60):

Downloads

Title	Date Published	Size	D/L
Advanced Debugging Techniques- Lab 1 of 3	11/29/2010 11:39:00 AM	21587 KB	
MATLAB Device Blocks for MPLAB IDE	5/5/2010 2:47:39 PM	36 KB	
MPASM/MPLINK User's Guide	4/8/2009 3:52:41 PM	2896 KB	
MPLAB Assembler, Linker and Utilities for PIC24 MCUs and dsPIC DSCs User's Guide	1/26/2010 10:16:32 AM	1981 KB	
MPLAB IDE Current Release Notes	12/23/2010 4:27:36 PM	252 KB	
MPLAB IDE User's Guide	1/20/2008 12:09:31 PM	4232 KB	
MPLAB IDE v8.63	12/27/2010 5:10:28 PM	113111 KB	
MPLAB IDE v8.63 PIC18F46K22 Family Debug Patch	1/24/2011 3:28:42 PM	452 KB	
Quick Guide to Microchip Development Tools	3/4/2011 10:09:50 AM	582 KB	
The MPLAB IDE Debug Tool API	5/13/2010 5:16:00 PM	171 KB	

Figura 60: Indicación 1-1.

Instalar en archivos de programa(Fig 61):

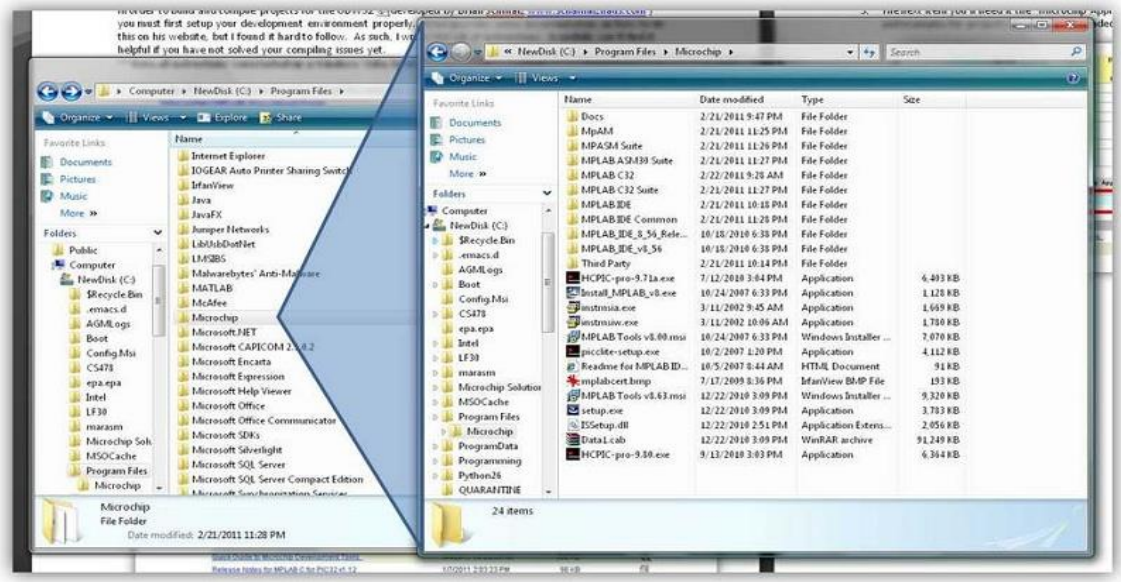


Figura 61: Indicación 1-2.

Nótese que hay las carpetas mplab c32 y mplab c32 suite.

2.-Ir a la página de microchip y descargar el instalador c-compiler.(Fig. 62)

Figura 62: Indicación 2.

Para que descargar, instalar y usar *c-compiler* se necesita una cuenta en microchip. La cuenta es gratuita.

Se instala en la misma carpeta que *mplab.*

3.- En la página de microchip descargar el M.A.L. (*Microchip aplications library*) que es un conjunto de librerías y ejemplos diseñados para ser cargados al PIC32 (Fig 63).

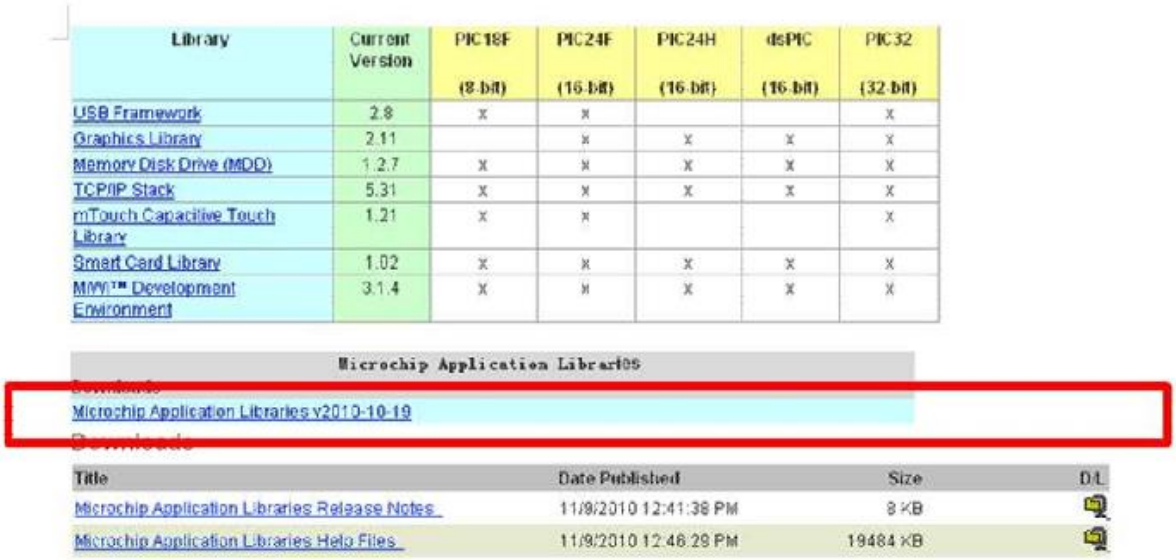


Figura 63: Indicación 3-1.

M.A.L. necesita ser instalado directamente en c: que es el directorio por defecto.

Una vez instalado se verifica que exista un directorio llamado microchip (Fig 64).

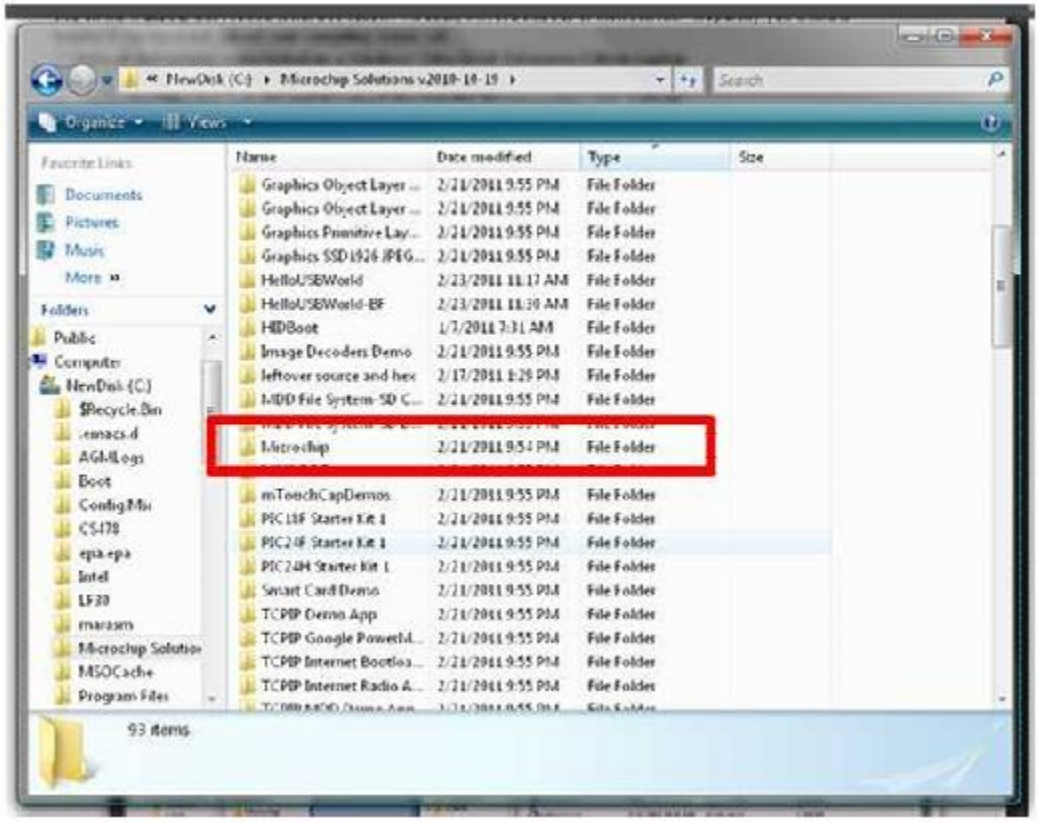


Figura 64: Indicación 3-2.

Todos los proyectos se guardaran en este directorio, para que las referencias de ubicación relativa funcionen.

4.- Descargar el ejemplo de la página del ubw32

http://www.schmalzhaus.com/UBW32/FW/HelloUSBWorld/HelloUSBWorld.zip

y extraerlo en el directorio mencionado anteriormente, este será un proyecto, después se abre el archivo “HelloUSBWorld.mcp” este es el tipo de archivo de proyecto principal que se abre con mplab.

5.- Una vez que se ha abierto mplab se va a configure->selectdevice y se asegura que sea el PIC correcto.

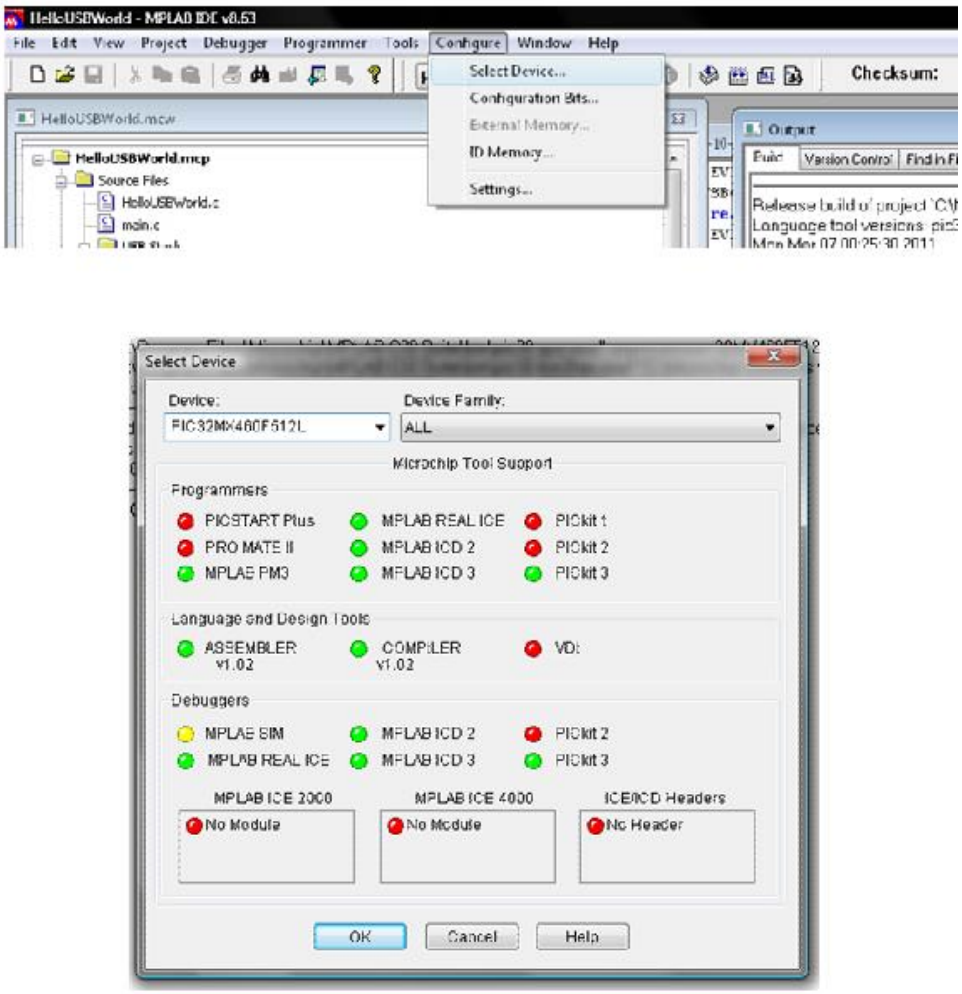


Figura 65: Indicación 5-1.

Ahora se va a programer->selectprogramer->selectnone.

Esto se utiliza para seleccionar el bootloader que trae integrado el PIC (Fig 66).

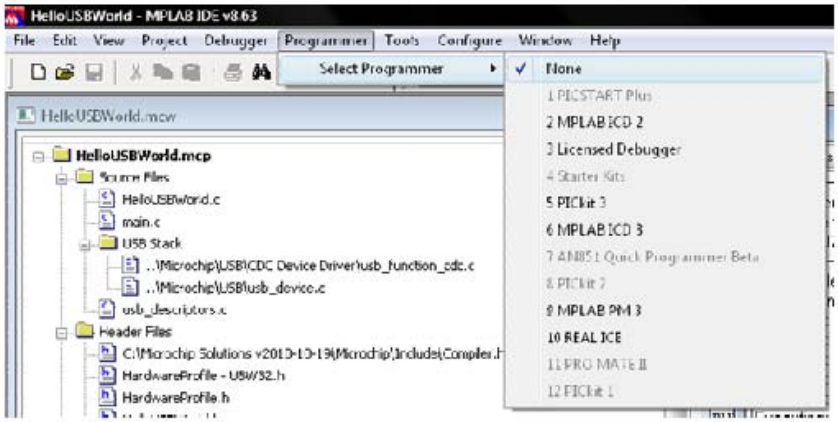


Figura 66: Indicación 5-2.

Ahora se compilara el programa, pare esto se pisa el botón buildall o las teclas ctrl+f10.

Puede que pregunte que compilador c (c-compiler) usara, hay que seleccionar la carpeta *mplab* c32 suite que se señala anteriormente como se muestra en la Figura 67.

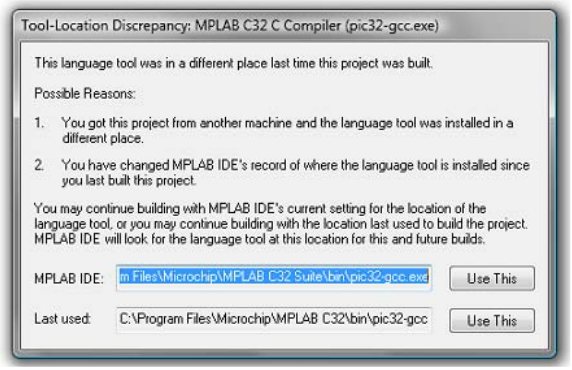


Figura 67: Indicación 5-3.

Hay que pisar el botón “use this”.

En caso de que falle, es porque *mplab* no tiene las referencias, en este caso hay que ir a Project->buildoptions-> Project, saldrá una ventana, después ir a la pestaña directories,y en el seleccionador marcar *include search path* y seleccionar C:\Program Files\Microchip\MPLAB C32 Suite\pic32mx\include (véase Figura 68).

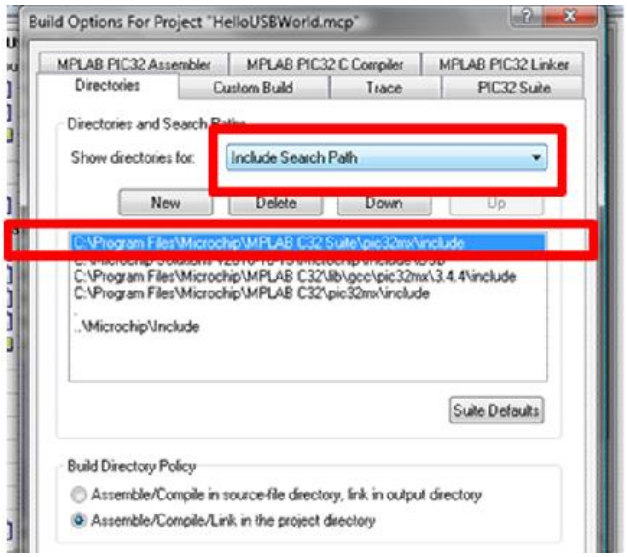


Figura 68: Indicación 5-4.

Si no marco error significa que se ha creado un archivo con terminación hex (Fig 69).

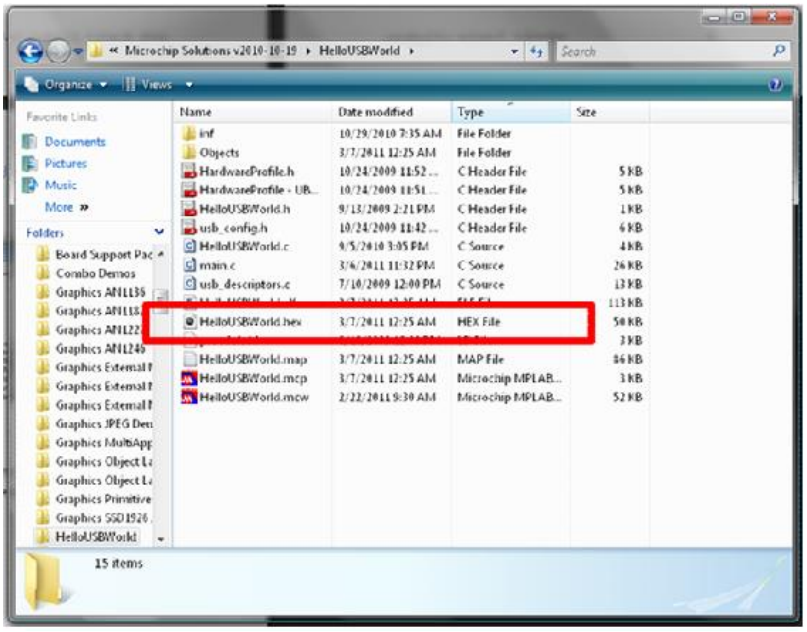


Figura 69: Indicación 5-5.

Apéndice 5 Cascos de prueba

Para que el casco original no sufriera ningún cambio y verificar que el Visor funcionara se hicieron algunos cascos en papel mache como modelo de pruebas (Fig 70).

Figura 70: Casco de Papel mache.

Los cascos se usaban para insertar el visor para ambos ojos y después se para un solo ojo.

Estos cascos de papel mache resultaron ser resistentes.

Apéndice 6 Cámara Web Genérica en Raspberry Pi

Como ya hemos mencionado que la Raspberry no acepta cualquier cámara ya sea USB o por el módulo de la cámara que está integrado.

Debido a que no hay un controlador compatible con *Raspbian*, para que una cámara común funcionase se descargaron los siguientes programas y librerías siguiendo los pasos a continuación:

- Instalación del programa *UvcCapture* y librerías v4linux:
  Para descargar los paquetes ejecutamos:
  sudo apt-get install uvccapture libv4l-0

- Instalación del programa ImageMagick:
  Sudo apt-get install imagemagick
  Instalamos este paquete de software para poder visualizar de una forma cómoda los ficheros obtenidos y poder "ejecutarlos" desde terminal (dentro del entorno gráfico).

- Conexionado y test hardware de la cámara USB:
  Para verificar si la cámara ha sido detectada como dispositivo Linux, se tecleo *ls /dev* lo que hará que aparezca en la terminal */dev/video0* si el sistema detecta la cámara como dispositivo.

- Prueba del sistema de vídeo:
  Con la cámara conectada y teniendo los programas adecuados ya instalados, se realiza una prueba de que todo está funcionando correctamente, para eso se toma una foto tecleando en la terminal:

  uvccapture -m -B50 -C10 –o "captura.jpg"

Cabe mencionar que se realizaron los pasos anteriores solo para comprobar que la cámara se puede utilizar y sin tener que estar en la lista de compatibilidad de Raspberry Pi.

Para ver video se realizó un programa en *Python.*

Apéndice 7 Auto Login en Raspberry Pi

Para que el casco fuese de inicio rápido y automático se modificó el sistema operativo de la siguiente forma para hacer que la Raspberry Pi tenga auto login.

Al iniciar la Raspberry en modo terminal (es más rápido y eficiente), escribimos lo siguiente:

sudo nano /etc/inittab

Buscamos la siguiente línea para modificarla:

1:2345:respawn:/sbin/getty 115200 tty1

El numero 115200 puede variar, y a continuación le añadimos el símbolo gato quedando de la siguiente forma:

#1:2345:respawn:/sbin/getty 115200 tty1

El símbolo # tiene como función hacer que ésa línea valga como comentario y no se active

Como paso siguiente añadimos debajo de la línea anterior lo siguiente:

1:2345:respawn:/bin/login -f pi tty1 </dev/tty1 >/dev/tty1 2>&1

De tal forma que indicamos a “pi” como el usuario por defecto y de esta forma se modifica el fichero inittab para hacer auto login.

Ahora guardamos presionando las teclas Ctrl+X.

Apéndice 8 Iniciar Python desde el Sector de Arranque

Para esto al igual que en otras plataformas basadas en debían se repiten algunos pasos.

Primero creamos un script en el archivo *init.d* el cual registra lo que debe iniciar al entrar al S.O. *Raspbian.*

Desde modo terminal en modo administrador o súper usuario (como se le conoce en Linux) se teclea:

sudo nano /etc/init.d/nombre_del_script

(camvid en nuestro caso).

Lo mejor es que el script tenga un nombre que no esté repetido en el directorio anterior o no se iniciara, y dañara el sistema operativo.

Al iniciar el editor de texto “nano” escribimos lo siguiente:

Suponiendo que el programa en *Python* se encuentra en /home/pi/Desktop (ruta del programa a utilizar).

```
#! /bin/sh
# /etc/init.d/tu_script

#para hacer que Python inicie y corra el programa (script) que queremos
python /home/pi/Desktop/tu_script_en_python.py

case "$1" in
start)
   echo "iniciando"
   # para correr la aplicación
   sudo /home/pi/Desktop/tu_script_en_python.py
   ;;
stop)
   echo "apagando"
   # matando el proceso de python
killallpython
   ;;
```

```
  *)
exit 1
  ;;
esac

exit 0
```

Después guardar presionando Ctrl+X, recordando que este guardado no es para siempre para ello se hace un script ejecutable dentro de init.d.

Entonces nos regresara a la línea de comandos.

Generando un script ejecutable

Hay que hacer el script ejecutable y para ello en la misma línea de comandos escribimos:

```
Sudo chmod 755 /etc/init.d/script
```

Para luego probar que funciona el script a iniciar en el sector de arranque (boot).

```
Sudo /etc/init.d/scriptstart
```

Para detener la prueba.

```
Sudo /etc/init.d/script stop
```

Ahora para hacer que cada vez que inicie el Raspberry Pi inicie el programa.

Escribimos:

```
Sudo update-rc.dt el_script defaults
```

Esto hará que permanezca en el registro y se inicie el programa.

Para quitarlo de estas aplicaciones de inicio:

```
sudo update-rc.d -f el_script remove
```

Apéndice 9 Modificación a la Mini PC

Basándonos en la idea de hacer que la Mini-PC encendiese rápido y se dirigiera a iniciar la cámara de manera automática sin intervención del usuario (así como sucedió en la RPi) se realiza lo siguiente:

Primero un programa que active la cámara del mini PC. Pero también detecta si hay otra cámara y activa la principal.

```
packagecom.project.smartstay.browserapplication;
importjava.io.IOException;
importandroid.app.Service;
importandroid.content.Context;
importandroid.content.Intent;
importandroid.graphics.Bitmap;
importandroid.graphics.BitmapFactory;
importandroid.hardware.Camera;
importandroid.hardware.Camera.CameraInfo;
importandroid.hardware.Camera.Parameters;
importandroid.media.AudioManager;
importandroid.media.FaceDetector;
importandroid.media.FaceDetector.Face;
importandroid.os.IBinder;
importandroid.util.Log;
importandroid.view.SurfaceHolder;
importandroid.view.SurfaceView;

public class CameraService extends Service implements SurfaceHolder.Callback{

    //a variable to store a reference to the Surface View at the main.xml file
privateSurfaceViewsv;
booleanpreviewing = false;
//se captura una imagen en mapa de bits
//Camaras
    //tsurfaceholder
privateSurfaceHoldersHolder;
//una variable para el control de la cámara
private Camera mCamera=null;
    //parametros de las camaras
```

```
privateParametersparameters;

    /** Se llama cuando se crea por primera vez. */
   @Override
Public void onCreate()
   {
super.onCreate();
Camera.CameraInfocameraInfo = new Camera.CameraInfo();
Log.d("No of cameras",Camera.getNumberOfCameras()+"");
for (intcamNo = 0; camNo<Camera.getNumberOfCameras(); camNo++) {

CameraInfocamInfo = new CameraInfo();
Camera.getCameraInfo(camNo, camInfo);

if (camInfo.facing==(Camera.CameraInfo.CAMERA_FACING_FRONT)) {
mCamera = Camera.open(camNo);
         }
      }
if (mCamera == null) {
        // No hay cámara frontal, se usa la camera trasera, pero dejamos esto dentro del
código por si se reconoce la camara como frontal

mCamera = Camera.open();
      }
sv = new SurfaceView(getApplicationContext());
Log.d("Entered","dfdf");
   }
   @Override
publicintonStartCommand(Intent intent, int flags, intstartId) {
Log.d("Entered onstart","dfdf");

//        Toast.makeText(getApplicationContext(), "wrking ", Toast.LENGTH_LONG).show();
try {

            //se establecen parámetros de la cámara
mCamera.setParameters(parameters);
mCamera.startPreview();
mCamera.takePicture(null, null, mCall);
```

```
        } catch (IOException e) {
            // TODO Auto-generado catch block
e.printStackTrace();
        }
sHolder = sv.getHolder();
sHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
returnsuper.onStartCommand(intent, flags, startId);
   }
Camera.PictureCallbackmCall = new Camera.PictureCallback()
   {

public void onPictureTaken(byte[] data, Camera camera)
{
         //se guarda el archivo en algún formato de imagen
FileOutputStreamoutStream = null;
try{
outStream = new FileOutputStream("/sdcard/Image.jpg");
outStream.write(data);
outStream.close();
            } catch (FileNotFoundException e){
Log.d("CAMERA", e.getMessage());
            } catch (IOException e){
Log.d("CAMERA", e.getMessage());
            }
        }
   };
    @Override
publicIBinderonBind(Intent intent) {
        // TODO Auto-generado method stub
return null;
    }
public void surfaceChanged(SurfaceHolder holder, int format, int width,
int height) {
if(previewing){
mCamera.stopPreview();
previewing = false;
        }
```

```
    }
public void surfaceCreated(SurfaceHolder holder) {
    }
public void surfaceDestroyed(SurfaceHolder holder) {
mCamera.stopPreview();
mCamera.release();
mCamera = null;
previewing = false;
    }
    }
```

Se crea el código que hacer que se ejecute un programa al iniciar el dispositivo

```
importandroid.app.Notification;
importandroid.app.NotificationManager;
importandroid.app.PendingIntent;
importandroid.app.Service;
importandroid.content.Context;
importandroid.content.Intent;
importandroid.net.Uri;
importandroid.os.IBinder;
importandroid.widget.Toast;

public class MyService extends Service {
private static MyService instance  = null;
public static booleanisRunning() {
return instance != null;
    }
    @Override
publicIBinderonBind(Intent intent) {
return null;
    }
    @Override
public void onCreate() {
Toast.makeText(getApplicationContext(), "ServicioMyServicecreado",
Toast.LENGTH_LONG).show();
System.out.println( "ServicioMyServicecreado");
instance=this;
    }
    @Override
```

```
public void onDestroy() {
Toast.makeText(getApplicationContext(), "ServicioMyServicedestruido",
Toast.LENGTH_LONG).show();
System.out.println( "ServicioMyServicedestruido");
instance = null;
    }
    @Override
public void onStart(Intent intent, intstartid) {
Toast.makeText(getApplicationContext(), "ServicioMyServiceiniciado!!",
Toast.LENGTH_LONG).show();
System.out.println( "Servicio MyService iniciado");
lanzarNotificacion();
}
voidlanzarNotificacion(){
      String ns = Context.NOTIFICATION_SERVICE;
NotificationManagernotManager = (NotificationManager) getSystemService(ns);
      //Configuramos la notification
      Notification notif = new Notification(android.R.drawable.ic_menu_agenda,
"MyService", System.currentTimeMillis());
//Configuramos el Intent
Context contexto = MyService.this;
CharSequencetitulo = "MyServiceNotification";
      //Intent que se abrira al clickear la notificacion
Intent resultIntent =new Intent(Intent.ACTION_VIEW,
Uri.parse("http://garabatoslinux.net"));;
PendingIntentcontIntent = PendingIntent.getActivity(contexto, 0, resultIntent, 0);
notif.setLatestEventInfo(contexto, titulo, descripcion, contIntent);
notif.flags |= Notification.FLAG_AUTO_CANCEL;
notif.defaults |= Notification.DEFAULT_VIBRATE;
notManager.notify(1, notif);
     }
   }
```

Tenemos que hacer que nuestra clase se extienda de Service y sobreescribir los métodospublicIBinderonBind(Intentintent) , publicvoidonCreate() , publicvoidonDestroy() y publicvoidonStart(Intentintent, intstartid).

Únicamente estos 4 métodos hay que sobreescribir, y podremos agregar todos los métodos que nosotros necesitemos.

```
<service android:enabled="true"
android:name="com.example.testmyservice.MyService"
android:icon="@drawable/ic_launcher" />
```

Una vez hecho esto, solo nos queda invocar el servicio.

Para que el servicio se inicie automáticamente al iniciar el sistema, agregamos las siguientes líneas a nuestro manifiesto dentro del tag<application>:

```
<receiver android:name="com.example.testmyservice.Receiver">
<intent-filter>
<action    android:name="android.intent.action.BOOT_COMPLETED" />
<category android:name="android.intent.category.HOME" />
</intent-filter>
</receiver>
```

Y agregamos el siguiente permiso:

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

Con esto, nuestra aplicación le está indicando al sistema que, al iniciarse, invoque a la clase com.example.testmyservice.Receiver, que es una clase que extiende de BroadcastReceiver y se define de la siguiente manera:

```
importandroid.content.BroadcastReceiver;
importandroid.content.Context;
importandroid.content.Intent;

public class Receiver extends BroadcastReceiver{
      @Override
public void onReceive(Context context, Intent intent) {
         Intent serviceIntent = new Intent(context,MyService.class);
if(!MyService.isRunning())context.startService(serviceIntent);
}
   }
```

Básicamente, lo que hace nuestra clase es ser invocada automáticamente al iniciar el sistema e iniciar nuestro servicio (la cámara).