CAMARA AINTERVALOS
TIME LAPSE
BAJO PRESUPUESO

CAMARA LLAVERO

EEstas diminutas mini camaras que han tomado muchas formas como llavero, paquete de goma de mascar, encendedor, mechero, boligrafo, etc basicamente constan un sensor de imagen CMOS, un procesador de imagen y una bateria que cumplen la funcion de grabar video (con audio!) y tomar fotografias, generalmente en una memoria extraible micro SD.

La calidad de la construccion de estas camaras y sus componentes es bastante pobre (su precio generalmente es de menos de 5 DOLARES / EUROS / LIBRAS) por lo que suelen fallar con regularidad, La bateria recargable como en muchos otros dispositivos electronicos suele ser la principal falla. Si se posee una de estas camaras en algun cajon, por que dejaron de funcionar por problemas en la bateria, se le puede dar una segunda oportunidad como camara de fotografias a intervalos o "time lapse" con componentes que probablemente se tienen a la mano de forma que el presupuesto a invertir puede ser muy bajo.

Las caracteristicas del proyecto serian las siguientes:

- Intervalo programable
- Muy bajo consumo de energia
- Flash automatico
- Bateria externa
- Bajo presupuesto (podria ser 0)

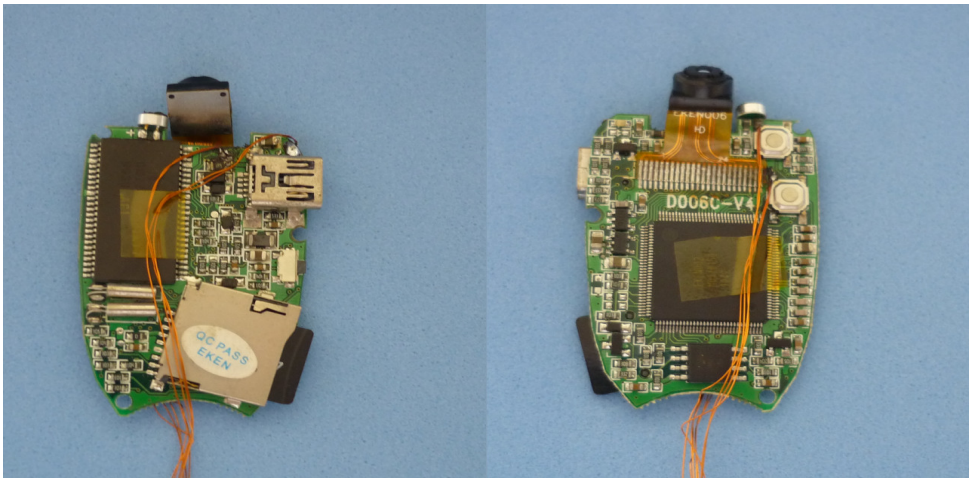
CONEXIONES EXTERNAS

Estas camaras cuentan con interruptores o pulsadores externos para habilitar sus diferentes modos, como la grabacion de videos o la toma de fotografias fijas. Generalmente se cuenta con un pulsador que al dejarse presionado por un determinado lapso de tiempo enciende la camara y si se presiona de nuevo por un determinado lapso de tiempo apaga la camara. Este apagado/ encendido de la camara es totalmente "por software" pues no se desconecta nunca la alimentacion.

El otro interruptor pulsador se utiliza como disparador para tomar fotografias fijas, si se pulsa y se suelta rapidamente tomara una fotografia, por el contrario si este interruptor se deja presionado por cierto intervalo de tiempo, se pondra en modo de grabacion de video continuo, y seguira grabando hasta que este boton sea presionado de nuevo o hasta que la bateria de la camara se agote. Si no se esta grabando video, la camara se apagara automaticamente despues de un determinado tiempo, sino se esta ejecutando ninguna accion como tomar fotografias.

La cantidad de interruptores, los intervalos para encender/apagar, tomar fotografias, iniciar/ detener grabacion de video pueden variar un poco dependiendo del fabricante y modelo exacto de la camara, pero en general, suelen funcionar como se menciono anteriormente.

Para controlar de forma externa la camara y para alimentarla, en caso de bateria defectuosa o si se requiere una mayor autonomia, se deberan soldar cables en el circuito impreso tanto en los pulsadores, como en los contactos de alimentacion. En lo posible usar cable aislado y del menor calibre posible.



AUTOMATIZANDO PULSACIONES

Para automatizar la toma de fotografias, basta con un simple un simple circuito ya sea mediante un integrado 555, transistores, o reles. Sinembargo si se requiere cosas mas complejas como tomar fotografias en intervalos variables, o en un intervalo mas largo que el tiempo de apagado de la camara o si se requiere algun tipo de "flash" activado cuando hay poca luz, se requerira de algun tipo de "inteligencia programable".

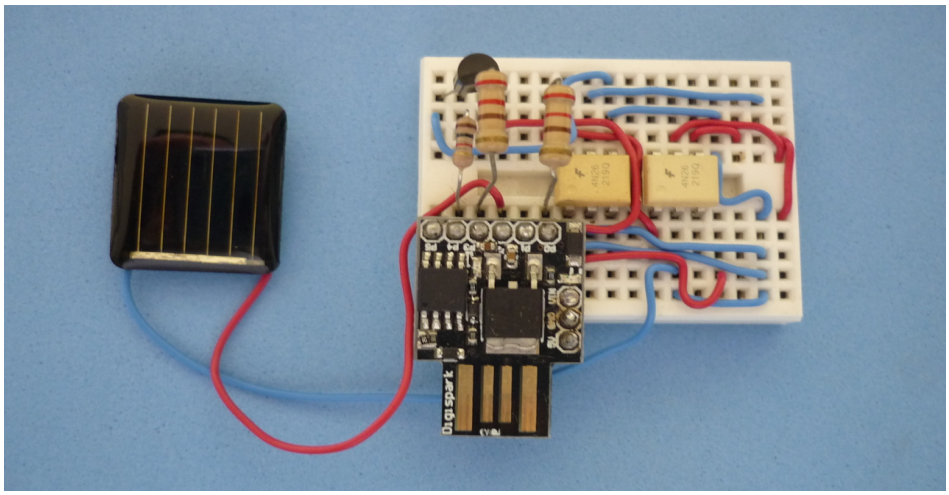
Para realizar esta tarea puede usarse cualquier sistema microcontrolado de preferencia, en este caso se usara un Digispark, el cual es un sistema muy pequeño, muy economico (cuesta aproximadamente 1 DOLAR/EURO/LIBRA con envio incluido a cualquier lugar del mundo), se conecta directamente al puerto USB sin requerir de un cable, ademas con un poco de esfuerzo puede hacerse compatible con la suite ARDUINO.

La secuencia ciclica que realiza el microcontrolador es la siguiente.

- Dejar presionado el pulsador de encendido de la camara y luego soltarlo. (Enciende camara)
- Esperar mientras se inicializa la camara
- Determinar las condiciones de iluminacion (mediante un panel solar miniatura usado en juguetes) para encender o no el "flash"
- Dejar presionado el interruptor de disparo de fotografia y soltar rapidamente (Para evitar entrar en modo de grabacion de video)
- Apagar el "flash" si es que este se encendio
- Esperar mientras se graba la fotografia en la memoria micro SD
- Dejar presionado el pulsador de encendido de la camara y luego soltarlo. (Apaga camara)
- Poner el microcontrolador en modo bajo consumo, activar el temporizador de watchdog.

Los diferentes tiempos de espera como pulsacion de botones, inicializacion, encendido, apagado, deberan ser hallados de forma experimental pues pueden variar dependiendo del tipo de camara usada. Generalmente uno de los interruptores funciona en modo "pull up" y el otro funciona en modo "pull down" verificar bien esto con un multimetro para determinar la forma de conexion de los optoacopladores.

El sistema se programo en C usando AVR GCC, y se deberan instalar los drivers Micronucleus para poder programar el Digispark



PROCESANDO IMAGENES

La camara almacenara las imagenes en la memoria Micro SD. En cada una de las imagenes se estampara la fecha y la hora que tenga programada la camara. Esto puede ser un problema si se quiere componer un video con todas las imagenes, pues no hay una forma de desactivar la estampacion de la fecha/hora en cada imagen. Existen algunas alternativas en el foro de 808, como modificar el firmware (solo para algunos modelos especificos) , aplicar filtros de video mediante virtual dub, etc.

Una alternativa simple para eliminar la estampa de fecha/hora en muchas imagenes, es utilizar Imagemagick, la idea es recortar de la foto la porcion donde aparece lo que se quiere eliminar, ademas puede ser necesario redimensionar para que la imagen cuadre con el tamaño del video de la siguiente forma:

mogrify -crop 1280x720+0+64 *.*

De esta forma se produce una imagen de 1280 x 720 pixeles como resultado, de donde se han recortado los ultimos 64 pixeles de la parte baja de la imagen original.

Por ultimo para generar un video de 30 fps a partir de multiples imagenes, que tengan un nombre en secuencia y empiecen desde cero (p.e EKEN0000.jpg, EKEN0001.jpg, EKEN0002.jpg ...), se puede usar avconv

avconv -r 30000/1001 -i EKEN%04d.jpg -r 30000/1001 video.mp4

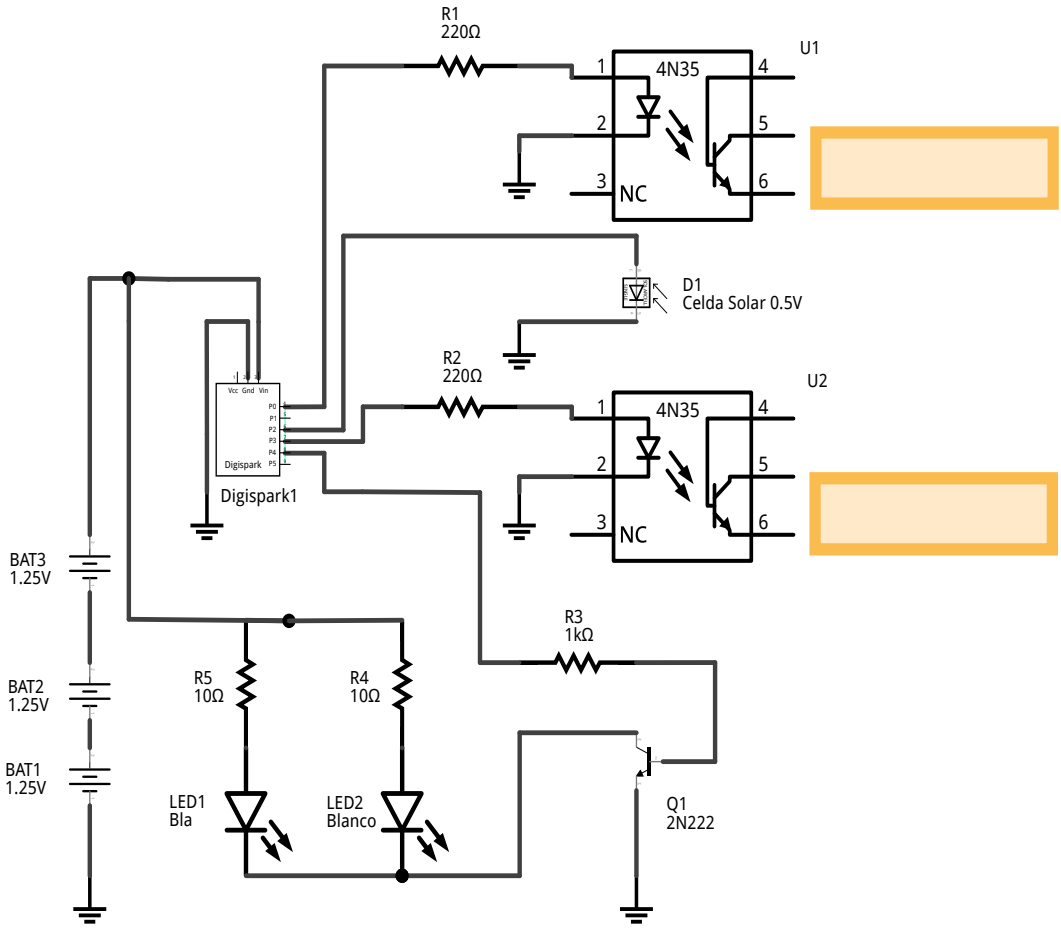
CONCLUSIONES

- La calidad de la imagen no es muy buena, por lo que se tiene que experimentar mucho con la distancia del objeto y la iluminacion para lograr resultados decentes.
- Poner el microcontrolador en modo bajo consumo y apagar la camara entre tomas, ahorra mucha energia, por lo que se puede utilizar baterias externas para tener un sistema relativamente "portatil"
- Existen muchas alternativas para toma de fotografias a intervalos time lapse, generalmente son mas costosas, o consumiran mas energia. Este simple ejercicio puede hacerse en par de horas, con elementos disponibles a la mano y con un bajo consumo de energia

DIAGRAMA ESQUEMATICO

Video del proceso de la camara y prueba de timelapse: **http://youtu.be/zPV5i8wKVYs**
Mini camara de grabacion digital: **http://s.click.aliexpress.com/e/N7MBqjqfu**
Panel solar miniatura: **http://s.click.aliexpress.com/e/ZjyR3rjAi**
Digispark: **http://s.click.aliexpress.com/e/rrR7UFQvj**

DIAGRAMA ESQUEMATICO



PROGRAMA DIGISPARK

main.c

```
// ***********************************************
//                                  Automatizanos.com
// ***********************************************
//  Includes
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
// ***********************************************

uint8_t conteo_watchdog = 0;

// ***********************************************
// Interrupciones
ISR(WDT_vect){
        ++conteo_watchdog;
}
// ***********************************************


#define ENCENDIDO_CAMARA            PB0
#define LED_INCORPORADO             PB1
#define SENSOR_ILUMINACION          PB2
#define DISPARO_CAMARA              PB3
#define LUZ_FLASH_EXTERNO           PB4



#define DELAY_DISPARO           1000
#define DELAY_CAM_LISTA 5000
#define DELAY_CAM_ENCE  2500
#define DELAY_CAM_FOTO  3000

int main(void) {

        // Init LED_INCORPORADO pin como salida
        DDRB |= (1 << LED_INCORPORADO);
        // Init ENCENDIDO_CAMARA pin como salida
        DDRB |= (1 << ENCENDIDO_CAMARA);
        // Init DISPARO_CAMARA pin como salida
        DDRB |= (1 << DISPARO_CAMARA);
        // Init LUZ_FLASH_EXTERNO pin como salida
        DDRB |= (1 << LUZ_FLASH_EXTERNO);

        // sequencia al reiniciar!! ( para determinar visualmente si ocurrio reset)

        PORTB |= (1 << LED_INCORPORADO);
        _delay_ms(100);
        PORTB ^= (1 << LED_INCORPORADO);
        _delay_ms(500);
        PORTB |= (1 << LED_INCORPORADO);
        _delay_ms(100);
        PORTB ^= (1 << LED_INCORPORADO);
        _delay_ms(1000);


        for (;;) {

                cli(); //Desabilitar interupciones
                MCUSR &= ~(1<<WDRF); // Limpiar el watchdog reset
                // desabilitar el temporizador watchdog
                WDTCR |= (1 << WDCE ) ;
                //WDTCR &= ( 0 << WDE );
```

```
WDTCR &= ~( 1 << WDE );

if(conteo_watchdog > 2){
	conteo_watchdog = 0;

	// *************************************************************************
	//                        ponga su codigo aqui!!
	// *************************************************************************

	// Encender  LED_INCORPORADO
	PORTB |= (1 << LED_INCORPORADO);

	_delay_ms(100);

	// presionando,esperando y soltando el interruptor para encender la camara
	PORTB |= (1 << ENCENDIDO_CAMARA);
	_delay_ms(DELAY_CAM_ENCE);
	PORTB &= ~ (1 << ENCENDIDO_CAMARA);

	// esperar a que la camara se inicialice
	_delay_ms(DELAY_CAM_LISTA);

	// Verificar las condiciones de luz
	inicializa_ADC();

	// iniciar medicion de  ADC
	ADCSRA |= (1 << ADSC);
	// Esperar mientras se completa la medicion
	while (ADCSRA & (1 << ADSC) );

	if (ADCH > 25)
		{
			// apagar el flash externo
			PORTB &= ~(1 << LUZ_FLASH_EXTERNO );
		} else {
			// encender el flash externo
			PORTB |= (1 << LUZ_FLASH_EXTERNO );
		}

	// presionando,esperando y soltando el interruptor de disparo de la camara
	PORTB |= (1 << DISPARO_CAMARA);
	_delay_ms(DELAY_DISPARO);
	PORTB &= ~ (1 << DISPARO_CAMARA);

	// Esperar a que la foto se grabe en la tarjeta
	_delay_ms(DELAY_CAM_FOTO);

	// Apagando el flash
	PORTB &= ~(1 <<  LUZ_FLASH_EXTERNO );

	_delay_ms(100);

	// presionando,esperando y soltando el interruptor para apagar la camara
	PORTB |= (1 << ENCENDIDO_CAMARA);
	_delay_ms(DELAY_CAM_ENCE);
	PORTB &= ~(1 << ENCENDIDO_CAMARA);

	_delay_ms(1000);

	// Apagar el led incorporado
	PORTB &= ~ (1 << LED_INCORPORADO);

	_delay_ms(100);

	// *************************************************************************
}

TCCR0B = (1<<CS01); // Poner el pre escalador en  8

MCUCR = (1<<SM1)|(1<<SE); // Inicializar modo de bajo consumo y habilitar dormir
```

```
                MCUSR &= ~(1<<WDRF); // Limpiar el  watchdog reset
                // Incializando el temporizador de watchdog
                WDTCR |= (1 << WDCE ) | ( 1 << WDE );
                WDTCR |= (1 << WDIE );
                WDTCR |= (1 << WDP3) | (1 << WDP0); // El temporizador se dispara cada 8 segundos

                sei(); //Habilitar la interrupciones

                asm volatile ("sleep");

        }

  return 0;
}

void inicializa_ADC()
{


        ADMUX =
                                        (1 << ADLAR) |   // Desplazar el resultado a la izquierda ( usar solamente 8 bits )
                                        (0 << REFS2) |    // Configura ref. de voltaje a 1.1V, bit 2
                                        (1 << REFS1) |    // Configura ref. de voltaje a 1.1V, bit 1
                                        (0 << REFS0) |    // Configura ref. de voltaje a 1.1V, bit 0

                                        (0 << MUX3) |    // usar ADC1 para entrada (PB2), MUX bit 3
                                        (0 << MUX2) |    // usar ADC1 para entrada (PB2), MUX bit 2
                                        (0 << MUX1) |    // usar ADC1 para entrada (PB2), MUX bit 1
                                        (1 << MUX0);     // usar ADC1 para entrada (PB2), MUX bit 0

        ADCSRA =
                                        (1 << ADEN) |    // Habilitar ADC
                                        (1 << ADPS2) |   // configura pre escalador a 128, bit 2
                                        (1 << ADPS1) |   // configura pre escalador a 128, bit 1
                                        (1 << ADPS0);    // configura pre escalador a 128, bit 0

}
```

MAKEFILE

makefile

```
# Automatizanos.com

DEVICE     = attiny85          # Ver avr-help para todos los dispositivos posibles
CLOCK      = 16000000          # 16Mhz
OBJECTS    = main.o            # Agregar mas objetos para cada uno de los archivos  .c aqui

UPLOAD = micronucleus --run
COMPILE = avr-gcc -Wall -Os -DF_CPU=$(CLOCK) -mmcu=$(DEVICE)

# objetivos simbolicos :
all:            main.hex

.c.o:
                $(COMPILE) -c $< -o $@

flash:          all
                $(UPLOAD) main.hex

clean:
                rm -f main.hex main.elf $(OBJECTS)

main.elf: $(OBJECTS)
                $(COMPILE) -o main.elf $(OBJECTS)

main.hex: main.elf
                rm -f main.hex
                avr-objcopy -j .text -j .data -O ihex main.elf main.hex
                avr-size --format=avr --mcu=$(DEVICE) main.elf
```