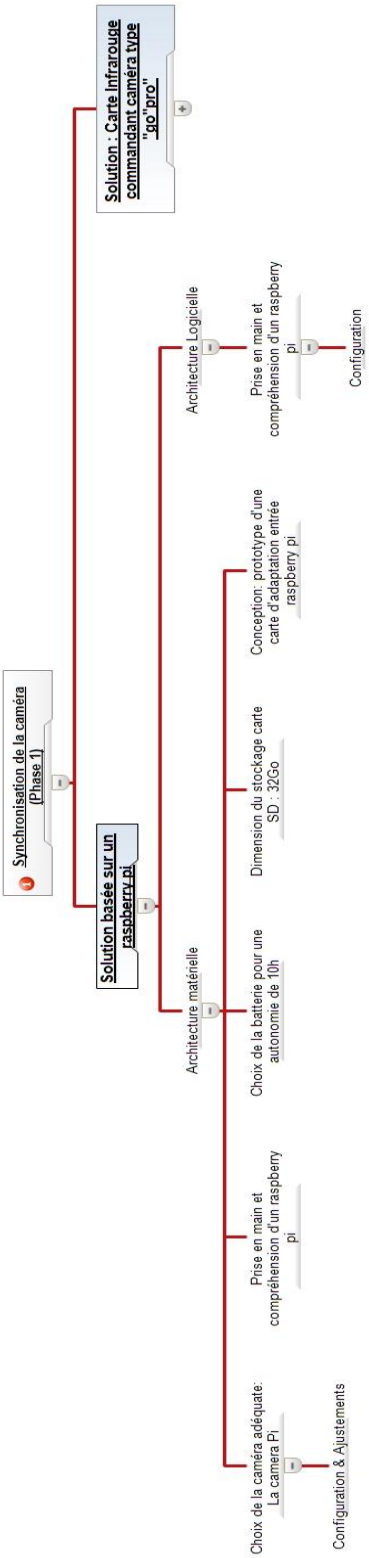
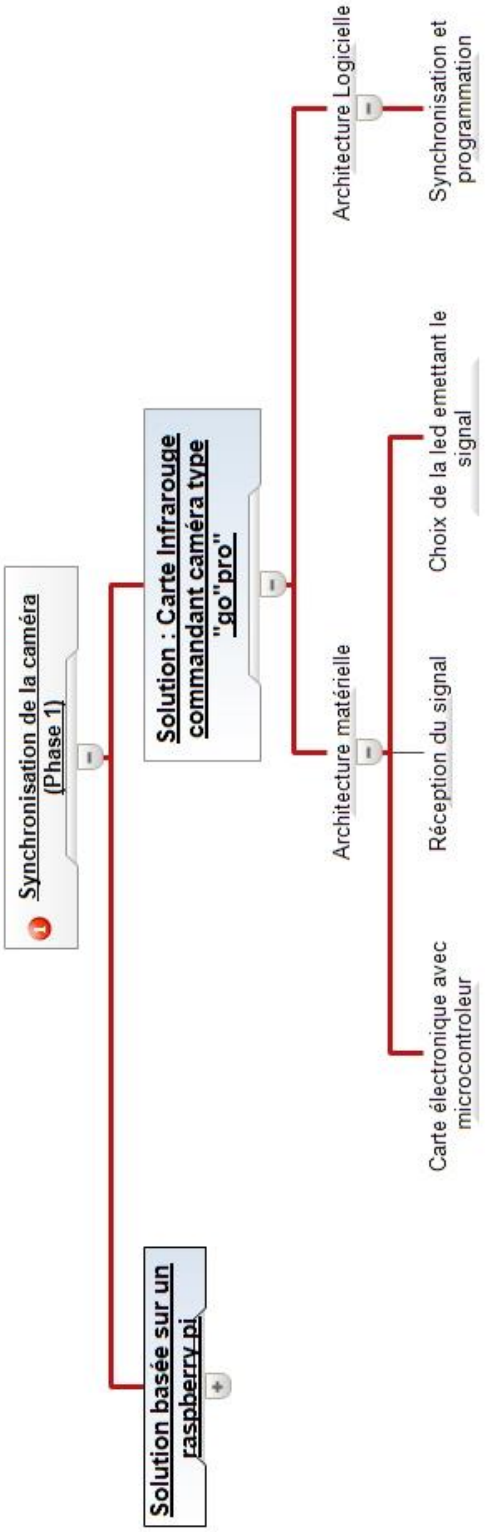
Table des annexes

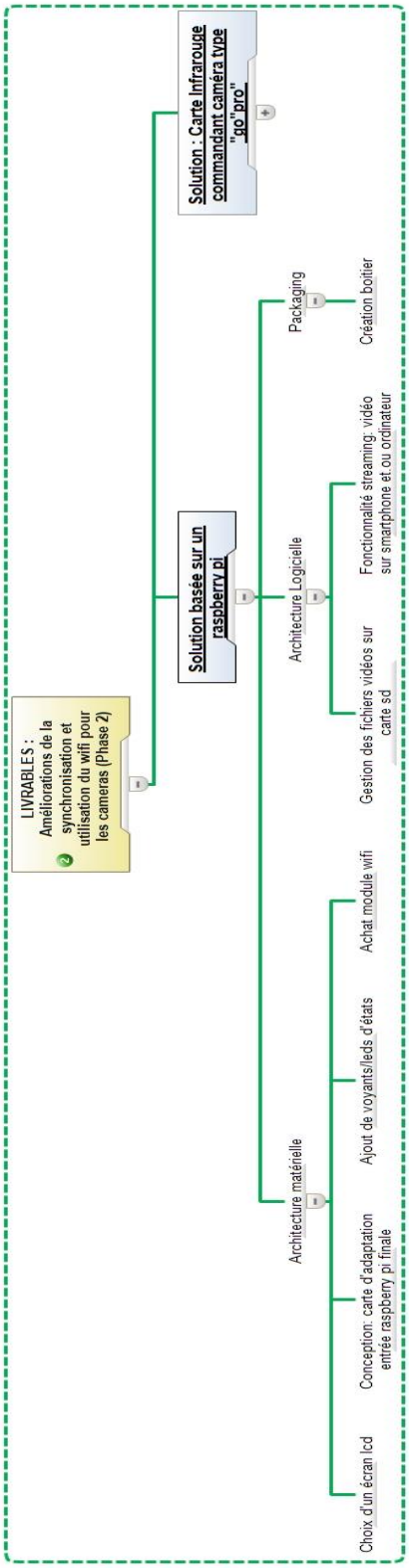
I. Développement de la Phase 1 : Solution basée sur un Raspberry Pi



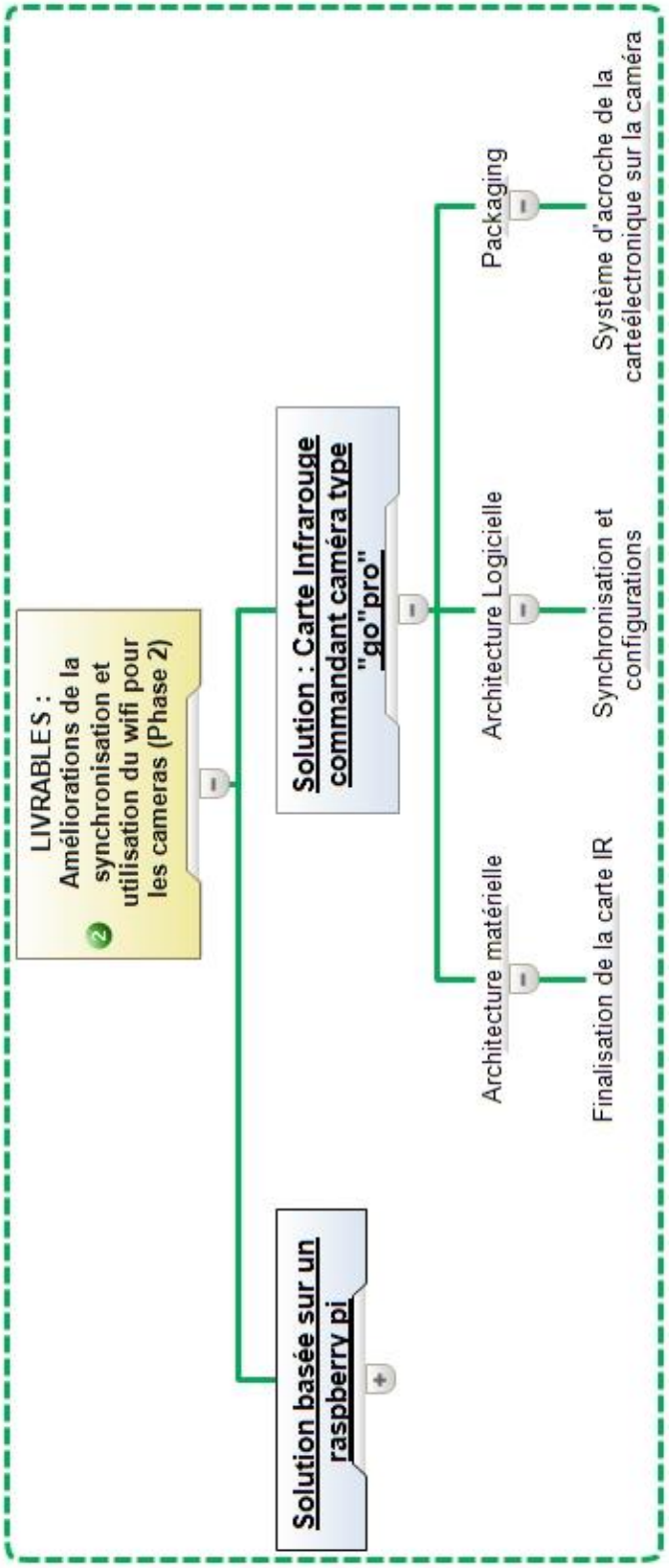
II. Développement de la Phase 1 : Solution basée sur une carte Infrarouge



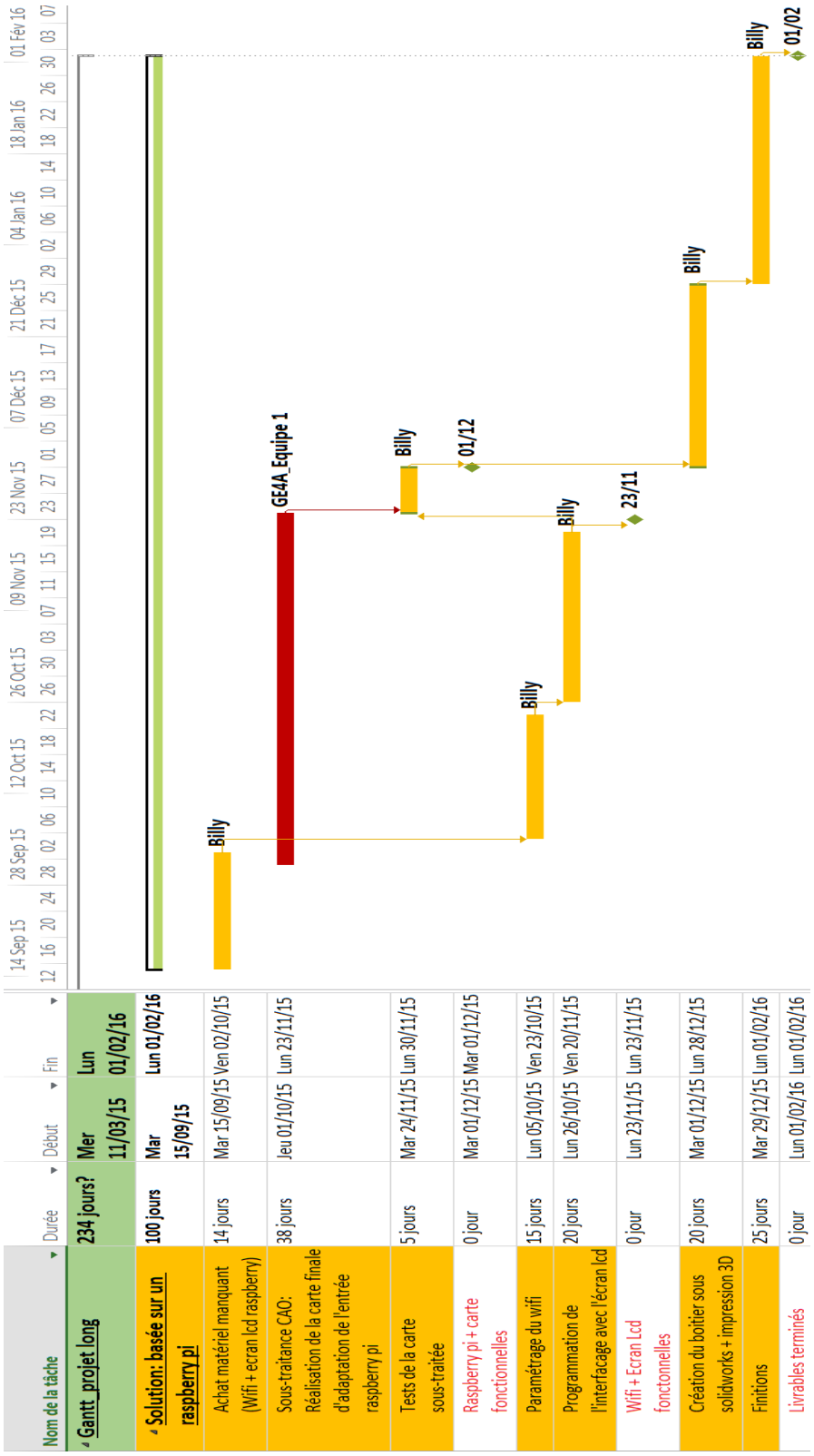
III. Développement de la Phase 2 : Solution basée sur un Raspberry Pi



IV. Développement de la Phase 2 : Solution basée sur une carte Infrarouge

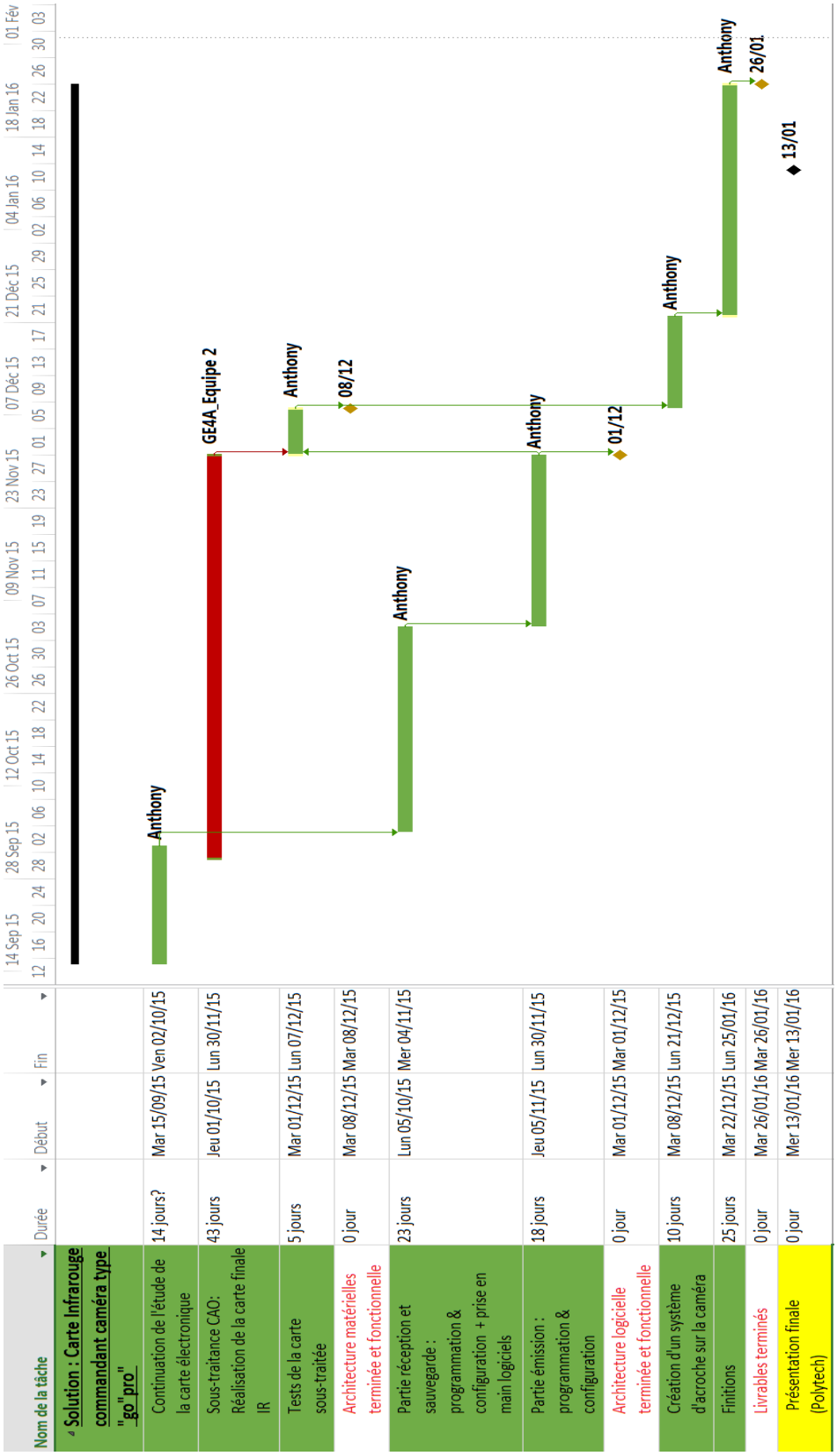


V. Planning de la phase 2 : Partie raspberry pi



Nom de la tâche	Durée	Début	Fin
Gantt_projet long	234 jours?	Mer 11/03/15	Lun 01/02/16
Solution: basée sur un raspberry pi	100 jours	Mar 15/09/15	Lun 01/02/16
Achat matériel manquant (Wifi + ecran lcd raspberry)	14 jours	Mar 15/09/15	Ven 02/10/15
Sous-traitance CAO: Réalisation de la carte finale d'adaptation de l'entrée raspberry pi	38 jours	Jeu 01/10/15	Lun 23/11/15
Tests de la carte sous-traitée	5 jours	Mar 24/11/15	Lun 30/11/15
Raspberry pi + carte fonctionnelles	0 jour	Mar 01/12/15	Mar 01/12/15
Paramétrage du wifi	15 jours	Lun 05/10/15	Ven 23/10/15
Programmation de l'interfacage avec l'écran lcd	20 jours	Lun 26/10/15	Ven 20/11/15
Wifi + Ecran Lcd fonctonnelles	0 jour	Lun 23/11/15	Lun 23/11/15
Création du boitier sous solidworks + impression 3D	20 jours	Mar 01/12/15	Lun 28/12/15
Finitions	25 jours	Mar 29/12/15	Lun 01/02/16
Livrables terminés	0 jour	Lun 01/02/16	Lun 01/02/16

VI. Planning de la phase 2 : Partie carte infra rouge



Nom de la tâche	Durée	Début	Fin
Solution : Carte Infrarouge commandant caméra type "go"pro"			
Continuation de l'étude de la carte électronique	14 jours?	Mar 15/09/15	Ven 02/10/15
Sous-traitance CAO: Réalisation de la carte finale IR	43 jours	Jeu 01/10/15	Lun 30/11/15
Tests de la carte sous-traitée	5 jours	Mar 01/12/15	Lun 07/12/15
Architecture matérielles terminée et fonctionnelle	0 jour	Mar 08/12/15	Mar 08/12/15
Partie réception et sauvegarde : programmation & configuration + prise en main logiciels	23 jours	Lun 05/10/15	Mer 04/11/15
Partie émission : programmation & configuration	18 jours	Jeu 05/11/15	Lun 30/11/15
Architecture logicielle terminée et fonctionnelle	0 jour	Mar 01/12/15	Mar 01/12/15
Création d'un système d'acroche sur la caméra	10 jours	Mar 08/12/15	Lun 21/12/15
Finitions	25 jours	Mar 22/12/15	Lun 25/01/16
Livrables terminés	0 jour	Mar 26/01/16	Mar 26/01/16
Présentation finale (Polytech)	0 jour	Mer 13/01/16	Mer 13/01/16

VII. Tableaux des coûts du projet

Désignation	Fournisseur	Prix unitaire TTC	Quantité
Raspberry Pi 2 model B	Kubii	39,99 €	2
Module Caméra pour Raspberry Pi 5MP	Kubii	22,86 €	2
Alimentation Raspberry PI B+ 5V 2A EURO	Kubii	8,90 €	2
Carte mémoire microSDHC SanDisk Ultra 32 Go Classe 10 + adaptateur SD (SDSDQUAN-032G-G4A)	Amazon	18,49 €	1
Carte mémoire microSDHC SanDisk Ultra 16 Go Classe 10 UHS-I + adaptateur SD (SDSDQUN-016G FFP-A)	Amazon	12,90 €	1
TeckNet PowerBank 9000mAh Batterie Externe Portable	Amazon	24,97 €	1
TOTAL TESTS PREQUALIFICATIS		199,86 €	
Led infrarouge	RadioSpare	quelques dizaines de centimes	
Dongle wifi	Amazon	20 €	
Exemple d'Ecran LCD 5 " :	Kubii	Estimation : 65€ + 5€ de frais de port	
Autres composants pour conception des différentes cartes	RadioSpare	quelques dizaines d'euros	

VIII. Code principal de la partie commande IR

VIII

```
/*  * File:   main.c
 * Author: Titi
 * * Created on 5 mai 2015, 17:11
 */

#include <p18f4550.h>
#include <stdio.h>
#include <stdlib.h>
#include "configuration.h"

void YourHighPriorityISRCode();
void YourLowPriorityISRCode();

unsigned short TIMER0_overflows, TIMER1_overflows;

int main(int argc, char** argv)
{
    //reglage de l'oscillateur
    OSCCONbits.SCS1 = 1;//internal oscillator selected
    OSCCONbits.SCS0 = 0;
    OSCCONbits.IOFS = 1;//INTOSC frequency is stable
    OSCCONbits.IRCF2 = 1;//8 MHz (INTOSC drives clock directly, Fosc=2MHz)
    OSCCONbits.IRCF1 = 1;
    OSCCONbits.IRCF0 = 1;

    RCONbits.IPEN = 1; //mise à 1 de IPEN "Enable priority levels on interrupts"

    INTCON = 0b11110000;     //GIE/GIEH=1, enables all high priority interrupts
                             //PEIE/GIEL,  enables all low priority interrupts
                             //TMR0IE,     enables the TMR0 overflow interrupt
                             //INT0IE,     enables the INT0 external interrupt(on RB0)
                             //RBIE,       disables the RB port change interrupt
                             //TMR0IF, TMR0 register did not overflow
                             //INT0IF, the INT0 external interrupt did not occur
                             //RBIF, none of the RB7:RB4 pins have changed state
```

```
    INTCON2bits.INTEDG0 = 1; //interrupt on rising edge on RB0
    INTCON2bits.TMR0IP = 1; //high priority for TMR0 overflow

    ADCON1bits.PCFG=0x0F;   //port in digital I/O

    T0CON=0b00000101;   //TMR0ON=0 "stops timer0"
                        //T08BIT=0 "Timer0 is configured as a 16-bit timer"
                        //T0CS=0 "internal instruction cycle clock (CLKO)
                        //T0SE=0 "increment on low-to-high transition on T0CKI pin"
                        //PSA=0 "Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output"
                        //T0PS2:T0PS0=101 "1:64 Prescale value" attention on a Fosc/4 en entrée du prescaler
                        //Fosc/4=2MHz => TIMER0 f=2e6/64=31250Hz

    //On veut une tempo de 2 sec avec TIMER0
    //2*(2e6/64)=62500
    //ce qui rentre dans 16 bits (max 65535)
    // /!\ le flag est mis à 1 après le passage de 65535 à 0
    //65535-62500+1=3036 (0BDC en hexa)
    TMR0H=0x0B;
    TMR0L=0xDC;

    //Init Timer1
    T1CONbits.RD16 = 1;     //read/write 16 bit values
    T1CONbits.T1CKPS1 = 0;  //1:1 prescale value
    T1CONbits.T1CKPS0 = 0;  //Ftimer1=Fosc/4/1=2MHz
    T1CONbits.T1OSCEN = 0;  //oscillator shut off
    T1CONbits.TMR1CS = 0;   //use internal clock (Fosc/4=2MHz)
    PIE1bits.TMR1IE = 1;    //enable timer1 interrupt on overflow
    PIR1bits.TMR1IF = 0;    //clear interrupt flag
    IPR1bits.TMR1IP = 1;    //high priority interrupt on TMR1 overflow

    //tempo de 50us avec timer1
    //50e-6*2e6=100
    //65535-100+1=65436 (0xFF9C)
    //on voit à l'oscilloscope qu'on a en fait 62us donc on enlève 12
    //38e-6*2e6=76
```

```
67
68      //tempo de 50us avec timer1
69      //50e-6*2e6=100
70      //65535-100+1=65436 (0xFF9C)
71      //on voit à l'oscilloscope qu'on a en fait 62us donc on enlève 12
72      //38e-6*2e6=76
73      //65535-76+1=65460 (0xFFB4)
74      TMR1H=0xFF;
75      TMR1L=0xB4;
76
77
78      //lecture du signal "début d'acquisition trame" par BP sur la PIN 0 du port B
79      TRISB=0b00000001; //RB0 en entrée
80      //voir TRISB page 118 et chapitre 9.2 9.9
81
82      TRISA = 0b00000000; //mise en sortie de tout le port A
83      PORTA =0;   // pour initialiser le port A à 0
84
85
86      /*INTCONbits.INT0IF = 1;  testent si les interruptions sur les flags marchent
87        INTCONbits.RBIF = 1;*/
88
89
90      while (1)
91      {
92          //permet de vérifier Fosc/4 de la clock
93          //avec Fosc/4=2MHz, RA0 est à 1 pendant 0,5us ce qui correspond bien à 2MHz
94          PORTAbits.RA0 = 1; // on met la broche RA0 à 1
95          PORTAbits.RA0 = 0; // on met la broche RA0 à 0
96
97      }
98
99      return 0;
00  }
```

```
103 #pragma code high_vector=0x08
104 void interrupt_at_high_vector(void) // Déclaration des interruptions à l'adresse 0x08
105 {
106     _asm goto YourHighPriorityISRCode _endasm   // Bouclage sur le programme d'interruption
107 }
108 #pragma code // Fin d'interruption, reprise du programme principal
109
110 #pragma interrupt YourHighPriorityISRCode
111 void YourHighPriorityISRCode()
112 {
113     if ( INTCONbits.INT0IF == 1 )//rising edge on RB0
114     {
115         TIMER0_overflows=0;
116         TIMER1_overflows=0;
117         PORTAbits.RA2 = 1; // on met la broche RA2 à 1
118
119         T1CONbits.TMR1ON=1; //enable timer1
120         T0CONbits.TMR0ON=1; //enable timer0
121
122         INTCONbits.INT0IF = 0;  //initialisation du flag
123     }
124
125     if ( INTCONbits.TMR0IF == 1 )//overflow on TMR0
126     {
127         TMR0H=0x0B;
128         TMR0L=0xDC;
129
130 /*code à utiliser pour allumer les leds pendant seulement deux secondes
131  * à remplacer ensuite par récupération et écriture du signal en mémoire
132
133         PORTAbits.RA2 = 0;  // on met la broche RA2 à 0
134         PORTAbits.RA4 = 0; // on met la broche RA4 à 0
135         T0CONbits.TMR0ON=0; //arrêt du TIMER0
136         T1CONbits.TMR1ON=0; //arrêt du TIMER1
137
138         TMR0H=0x0B;     //réécriture des registres pour une prochaine acquisition
```

```
141         TMR1H=0xFF;     //réécriture des registres pour une prochaine acquisition
142         TMR1L=0xB4;
143
144         INTCONbits.TMR0IF = 0;  //mise à 0 du flag
145
146
147 */
148
149         if(TIMER0_overflows%2 == 0)
150         {
151             PORTAbits.RA2 = 1; // on met la broche RA2 à 1
152         }
153         else
154         {
155             PORTAbits.RA2 = 0; // on met la broche RA2 à 0
156         }
157
158         TIMER0_overflows++;
159         if(TIMER0_overflows == 10000)
160             TIMER0_overflows = 0;
161
162         INTCONbits.TMR0IF = 0;  //initialisation du flag
163     }
164
165     if ( PIR1bits.TMR1IF == 1 )//overflow on TMR1
166     {
167         TMR1H=0xFF;
168         TMR1L=0xB4;
169
170         if(TIMER1_overflows%2 == 0)
171         {
172             PORTAbits.RA4 = 1; // on met la broche RA4 à 1
173         }
174         else
175         {
176             PORTAbits.RA4 = 0; // on met la broche RA4 à 0
177         }
178
179         TIMER1_overflows++;
180         if(TIMER1_overflows == 10000)
181             TIMER1_overflows = 0;
182
183         PIR1bits.TMR1IF = 0;  //initialisation du flag
184     }
185 }
186
187
188 #pragma code low_vector=0x18
189 void interrupt_at_low_vector(void)
190 {
191 _asm goto YourLowPriorityISRCode _endasm
192 }
193 #pragma code // Fin d'interruption, reprise du programme principal
194
195 #pragma interruptlow YourLowPriorityISRCode
196 void YourLowPriorityISRCode(){}
```

IX. Script principal pilotant la PiCamera sur le raspberry pi 2

```python
import time
import picamera
import datetime as dt
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setup(22, GPIO.IN, GPIO.PUD_DOWN)

with picamera.PiCamera() as camera:
    # camera.start_preview()
     camera.annotate_background = picamera.Color('Black')
     camera.annotate_text = dt.datetime.now().strftime('%d/%m/%Y - %H:%M:%S')
     for i in range(1, 9999):
         GPIO.wait_for_edge(22, GPIO.RISING)
         camera.start_recording('/boot/_Videos/Video_#%d.h264' %i)
         camera.wait_recording(0.2)
         GPIO.wait_for_edge(22, GPIO.FALLING)
         time.sleep(0.1)
         camera.stop_recording()

    # camera.stop_preview()
```

X. Script pour le bouton stop du raspberry pi 2

```python
# -*- coding: utf-8 -*-

import time
import os
import RPi.GPIO as GPIO

# Utilisation de la notation BCM (numero des pins)
GPIO.setmode(GPIO.BCM)

# Initialisation du BCM 23
GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Définition de la fonction arrêt
def stop(channel):
      # Affiche le message de confirmation
      print("Bouton arret appuye... Arrêt du Raspberry pi")
      # Réinitialisation du GPIO
      GPIO.cleanup()
      # Commande d'arrêt
      os.system('sudo halt')

# Attente d'enclenchement du bouton arrêt
GPIO.add_event_detect(23, GPIO.FALLING, callback=stop)

# Boucle infinie afin de garder le script actif
while 1:
      # Réduction de charge du CPU
      time.sleep(0.02)

# Réinitialisation des ports GPIO en sortie de script
GPIO.cleanup()
```

XI. Schéma et CAO de la carte prototype de régulation d'entrée pour le raspberry pi