|  |  |  |  |
| --- | --- | --- | --- |
|  | **Control de versión Investigación de Tecnologías** | FCV DIT | |
| Versión: 0 | R-DIT\_INST - 001 |
| PROCESO: **DIT - Investigaciones** | Página 1 de 1 | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Aplicativo:** | Librería Pulso Cardiaco | **Versión:** | 1.0 | **Fecha:** | 30/03/2020 |
| **Nombre del archivo fuente:** | | PulsoCardiacoFCV | | | |
| **Ruta archivo fuente:** | | GitHub\Hospital-smart-bracelet\Pulsera\Arduino\Librerias | | | |
| **Realizado por:** | | Cesar Augusto Figueroa Bayona | | | |
| **Resumen versión:** | | Creación de la librería Arduino PulsoCardiacoFCV. Esta librería se creó para la lectura de pulso cardiaco utilizando el sensor SEN0203.  En el archivo **PulsoCardiacoFCV.cpp** está la programación con las funciones para la lectura del pulso cardiaco con el Sensor SEN0203 usando un microcontrolador ATMega328p.  **Librerías Externas:**  #include "Arduino.h"  #include <Ticker.h>  **Variables:**  int \_pinSensorPPM  int \_pinVccSensorPPM  #define \_constanteLectura 15000 //Tiempo establecido para tomar la cantidad de pulsos.  int \_conteoPulsos = 0  int \_pulsosAnterior = 0  bool banderaControl = false  **Funciones:**  PulsoCardiacoFCV(int \_pinSensor, int \_pinVccSensor): Trae la información de los pines Vcc y de lectura del sensor SEN0203 y define el tipo de pin.  void FinCheck(): Realiza el conteo total de pulsos en un periodo de tiempo, si el conteo es menos a 10 entonces deja por defecto la lectura anterior.  void ReadPulso(): Lee el los datos del sensor y establece cuando hay un pulso, tiene en cuenta el periodo de datos leídos los cuales son de 0 a 1023, se considera un pulso cuando se realiza un cambio de 0 a 1023 y de 1023 a 0.  void StartSensorPPM(): Enciende el pin del sensor SEN0203 e inicia el ticket de lector de pulso.  void LoopSensorPPM(): Establece y mantiene el Loop de los tickers.  void StopSensorPPM(): Detiene los ticker y apaga el pin Vcc del sensor SEN0203  boolean isBanderaPrimerPulso(): Define cuando se realiza el primer pulso.  int ValorSPPM(): Calcula el PPM teniendo en cuenta los pulsos tomados en un periodo de tiempo (15 segundos) y retorna la variable con el dato del PPM.  **Tickers**  Ticker CheckPulso(FinCheck, \_constanteLectura): Establece el tiempo de duración de la función FinCheck.  Ticker LectorPulso(ReadPulso, 20): Establece a 20 ms el tiempo de cada lectura de sensor SEN0203.  En el archivo **PulsoCardiacoFCV.h** se establecen las variables públicas y privadas de la librería.  public:    PulsoCardiacoFCV(int \_pinSensor, int \_pinVccSensor)  void LoopSensorPPM()  void StartSensorPPM()  boolean isBanderaPrimerPulso()  int ValorSPPM()  void StopSensorPPM()  private:    void FinCheck()  void ReadPulso()  boolean banderaPrimerPulso = false  int \_conteoPulsos = 0  int \_pinSensorPPM = 0  boolean cambioEstado = false  int \_pinVccSensorPPM  En el archivo **Pulso\_Cardiaco\_FCV.ino** es el archivo principal donde se trabajan los archivos .cpp y .h creados anteriormente.  **Librerías:**  #include "PulsoCardiacoFCV.h"  **Variables:**  #define pinLecturaSensorppm A0 //Puerto de lectura del sensor de pulso cardiaco  #define pinVccSensorppm 7 //Puerto de lectura del sensor de pulso cardiaco  #define intervaloPPM 2000  int sppm = 0  unsigned long TSppm = 0  **Constructor:**  PulsoCardiacoFCV pulsocardiaco (pinLecturaSensorppm, pinVccSensorppm): Envía los datos de los pines Vcc y de lectura del Sensor SEN0203.  **Funciones:**  void ControlsensorPPM(): Función que ejecuta 4 procesos los cuales son:   * El primer proceso enciende el pin Vcc del sensor SEN0203 y establece el tiempo de ejecución del próximo proceso en 16000 ms. * El segundo proceso lee el dato PPM entregado por la librería y apaga el pin Vcc del sensor SEN0203 y establece el tiempo de ejecución del próximo proceso en 10 s. * El tercer proceso detiene los tickers y y establece el tiempo de ejecución del próximo proceso en 10 s. * El cuarto proceso imprime el valor PPM leído y establece el tiempo de ejecución del próximo proceso en 10 s.   void setup(): Establece el la velocidad de comunicación serial, define los puertos análogos para que tomen de referencia el voltaje interno del microcontrolador ATMEGA328P al cual corresponde a 1.1v con el comando analogReference(INTERNAL) e imprimir la finalización del Setup.    void loop(): Inicia el Loop de los ticker de la librería y ejecuta el ticker cada 2 segundos.  **Ticker:**  Ticker TikerControlSPPM(ControlsensorPPM, 10, 4, MILLIS): Ejecuta la función ControlsensorPPM 4 veces con los siguientes intervalos de tiempos:   * Primer intervalo de 10ms * Segundo intervalo de 16000ms * Tercer intervalo 10 ms * Cuarto intervalo 10ms   El archivo **keywords.txt** define el color de las variables de la siguiente forma:  PulsoCardiacoFCV KEYWORD1  ValorSPPM KEYWORD2  StartSensorPPM KEYWORD2  LoopSensorPPM KEYWORD2  isBanderaPrimerPulso KEYWORD2  StopSensorPPM KEYWORD2  El archivo **library.properties**    name=PulsoCardiacoFCV  version=1.0  author=FCV  maintainer=FCV  sentence=library for SEN0203  paragraph=library for SEN0203  category=Sensors  url=www.fcv.org  architectures=\*  depends=Ticker.h, Arduino.h  El archivo **README.md** establece la compatibilidad de la librería con el microcontrolador Atmega328p.  La librería se encuentra comprimida en un archivo .zip con todos los archivos mencionados anteriormente. Adicional se adjunta código SHA1 para la comprobación de los archivos adjuntos en el paquete .zip:  **Código SHA1**  d84ac10616b373182b69c23c33f1c990dda639e2  URL de la página SHA1  <https://emn178.github.io/online-tools/sha1_checksum.html> | | | |
| **Archivos modificados:** | | N.A. | | | |

***Elaborado Por:***

***\_Cesar Augusto Figueroa Bayona***

***Investigador en Tecnologías***

***Aprobado Por:***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***Director Infraestructura Tecnológica***