**Instituto Tecnológico y de Estudios Superiores de Monterrey**



**Campus Monterrey**

**Proyecto Integrador de Electrónica, Computación e Información**

**(TI-2006)**

**Reporte final proyecto RH-NOMA**

**Profesores:**

**Dra. Lorena Gómez**

**M.C. Luis Ricardo Salgado**

**Equipo 2**

**Alejandro Limones 803199**

**Andrés Martínez 920037**

**Jorge Saldívar 1033317**

**14 de Mayo de 2012**

**Índice:**

Introducción 3

Diagrama de arquitectura general 4

Requerimientos Funcionales. 5

Descripción de las pruebas de aceptación 5

Diagrama de casos de uso 6

Descripcion de casos de uso 7

Diagrama de clases 9

Modelo relacional 9

Diagramas de Secuencia 10

Diagramas de actividad 12

Diagrama de deployment 14

Anexos 15

Escenarios 15

Instancia del modelo relacional 16

Pruebas unitarias y de integración 17

Codigo de la aplicación impreso y documentado 20

Código Lenguaje C-Aplicación 20

Código Lenguaje C- Administrativo 38

Código Arduino – Conectado a lectoras RFID, Protocolo Wiegand 47

Código Arduino- conectado a base de datos. 53

Ejemplos de todos los reportes con datos significativos 54

# Introducción

El sistema desarrollado tiene como funciones principales automatizar el cáculo de nómina de los empleados y restringir su acceso a las instalaciones, basándose en permitir a cada trabajador entrar en ciertas puertas, días y horarios.

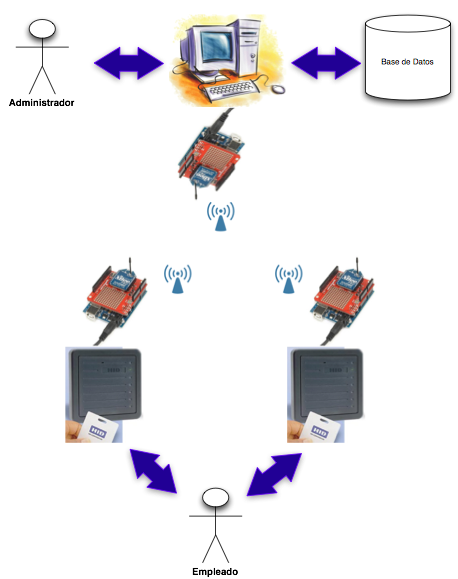
En sí el proceso da inicio cuando la lectora lee una tarjeta a través del arduino, la información de la tarjeta leida es enviada a otro arduino que esta conectado a una base de datos. En esta base de datos se tiene información del empleado, información de las entradas y salidas, puertas a la que el empleado tiene acceso y horarios en que se puede entrar y salir de las instalaciones de lunes a viernes, en sábados y domingos, con dicha información se puede dar o no acceso al empleado.

Al final de cada día se calculan las horas diarias trabajadas, con esto se pueden calcular las horas trabajadas y las extras diarias. Con esta información al final de cada quincena se puede calcular el ingreso por horas trabajadas y extras, el ingreso neto, los impuestos correspondientes y así el pago total por nómina del periodo de cada empleado.

El sistema fue desarrollado de la siguiente manera:

* La base de datos utilizada esta desarrollada en Access.
* La comunicación de los arduinos con la base de datos fue programada en lenguaje C.
* La comunicación entre los arduinos es inalámbrica a travez de Xbee.
* Las lectoras de proximidad RFID se comunican con los arduinos a través del protocolo Wiegand.

# Diagrama de arquitectura general

****

# Requerimientos Funcionales.

**Empleados**

1. Entrar al edificio.
2. Salir del edificio.

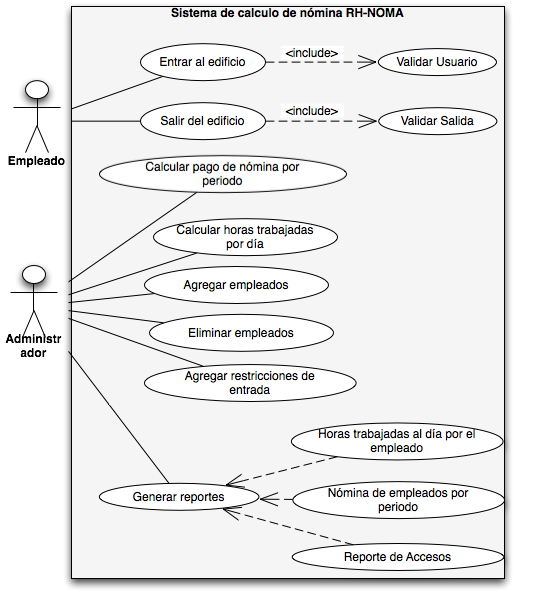
**Administrador**

1. Calcular pago de nómina.
2. Calcular horas trabajadas por día.
3. Agregar Empleados.
4. Eliminar Empleados.
5. Agregar restricciones de entrada a los empleados.
6. Generar reportes:
   1. Reporte de horas trabajadas al día por empleado.
   2. Reporte de Nómina del empleado por periodo.
   3. Reportes de accesos.

# Descripción de las pruebas de aceptación

1. Se le permite el acceso a un empleado, del cual se conoce su id de empleado, tiene acceso permitido a la puerta deseada, tiene permiso para entrar a la hora del día del intento.
2. Se le niega el acceso a un empleado, si no esta registrado, no tiene acceso a la puerta deseada o no se le permite la entrada a la hora del día deseado.
3. Para conocer si es entrada o salida, se busca un registro de Asistencia al cual le falte la hora de salida, si existe uno es una salida, se registra la fecha y hora y se calculan las horas trabajadas. Si no existe dicho registro, es una entrada. Al ser entrada se crea un nuevo registro con fecha y hora de entrada y el espacio de salida se deja pendiente.
4. Al final de cada día, se calculan las horas trabajadas y horas extra totales del día de cada empleado.
5. Al final del periodo, se calculan la nómina de cada empleado, del cual se obtienen sus horas trabajadas y las extras. Dichas cantidades se multiplican por el el salario y el salario de horas extra, con lo que se obtiene el salario neto, a este se le restan los impuestos y así se obtiene el pago total del periodo de cada empleado.

# Diagrama de casos de uso



# Descripcion de casos de uso

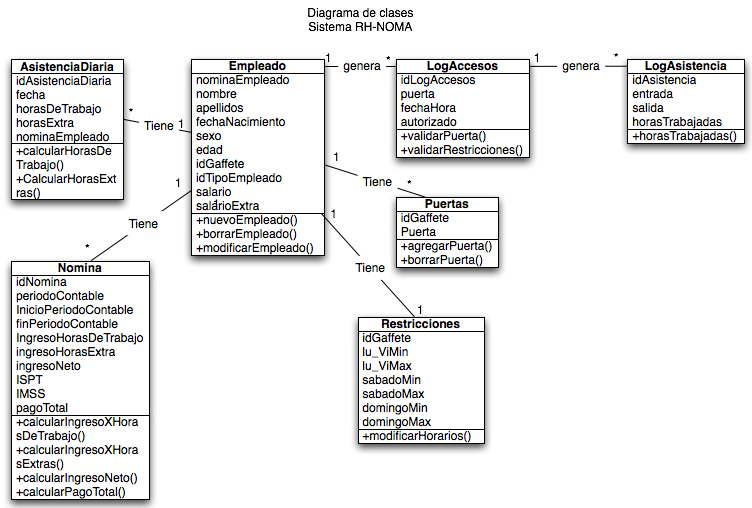
Documentación casos de uso 1

|  |  |
| --- | --- |
| **ID**: UC1 | **Use Case**: Entrar al edificio. |
| **Actors**: Empleado. | |
| **Purpose**: Validar que el empleado tiene acceso al edificio. | |
| **Preconditions**: Que la hora de entrada este en el rango válido. | |
| **Flow of events:** | |
| **Actor Action**  1.- El empleado pasa su credencial por la lectora.  6.- El empleado entra. | **System Reaction**  2.- Registra los datos de usuario que esta intentando ingresar.  3.- Valida que el usuario tenga acceso en la puerta.  4.- Valida que el usuario se encuentra en horario de entrada.  5.- Crea un registro de entrada con los datos del empleado. |
| **PostConditions**: Al usuario se le permite entrar al sistema. | |
| **Exceptions:**  -En caso de que el usuario no tenga acceso a esa puerta, se le niega la entrada.  -En caso de que el usuario no esté en horario válido para entrar, se le niega la entrada. | |

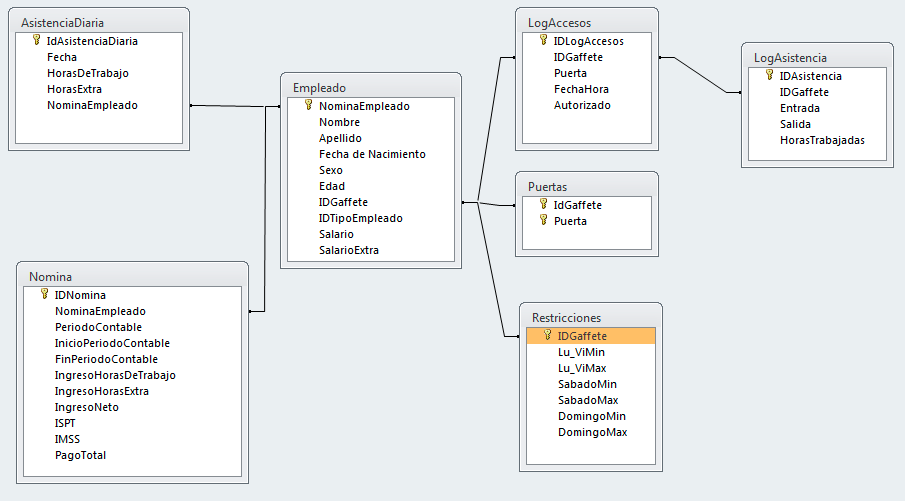
Documentación casos de uso 2

|  |  |
| --- | --- |
| **ID**: UC2 | **Use Case**: Calcular pago de nómina por periodo |
| **Actors**: Administrador | |
| **Purpose**: Calcular el pago de nómina de cada usuario por periodo. | |
| **Preconditions**: Empleado haya trabajado en la quincena y tenga registros de Asisitencia | |
| **Flow of events:** | |
| **Actor Action**  1.- El administrador entra al sistema  2.- Presiona la opción de generar nómina del periodo. | **System Reaction**  3.- Calcula las horas de trabajo y las extras trabajadas del periodo. 4.- Calcula el ingreso por horas de trabajo. 5.- Calcula el ingreso por horas extra. 6.- Calcula el ingreso neto(horas trabajadas + horas extra). 7.- Descuenta los impuestos. 8.- Calcula el pago total 9.- Guarda el calculo de nómina. |
| **PostConditions**: Queda calculado el salario de cada empleado del periodo. | |
| **Exceptions:**  -En caso de que el usuario no tenga horas extras, sigue con el paso 6. | |

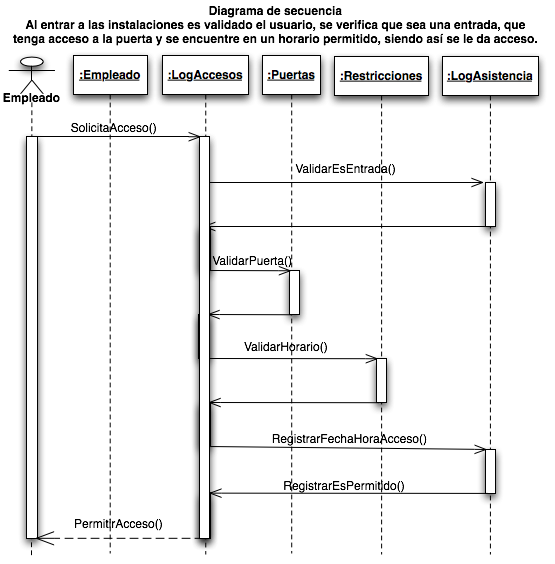
# Diagrama de clases

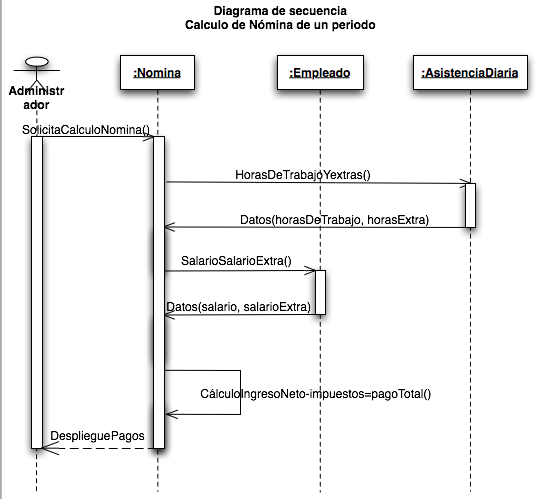


# Modelo relacional

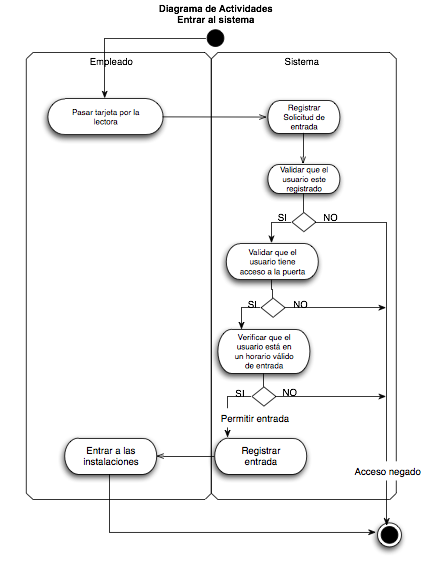


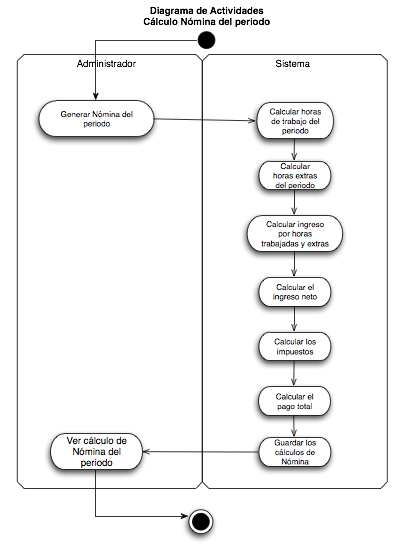
# Diagramas de Secuencia

****

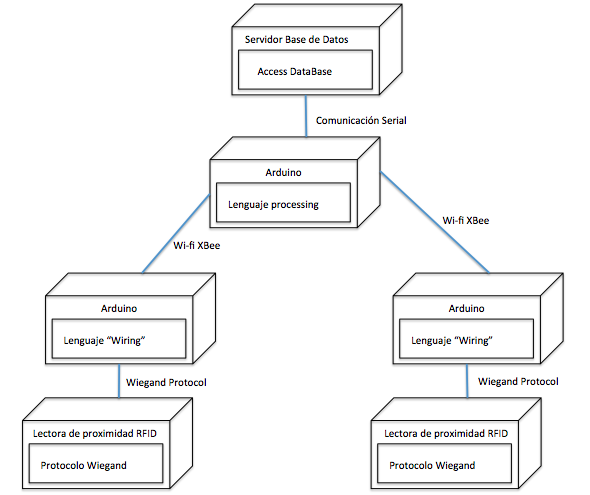
****

# Diagramas de actividad

****

****

# Diagrama de deployment

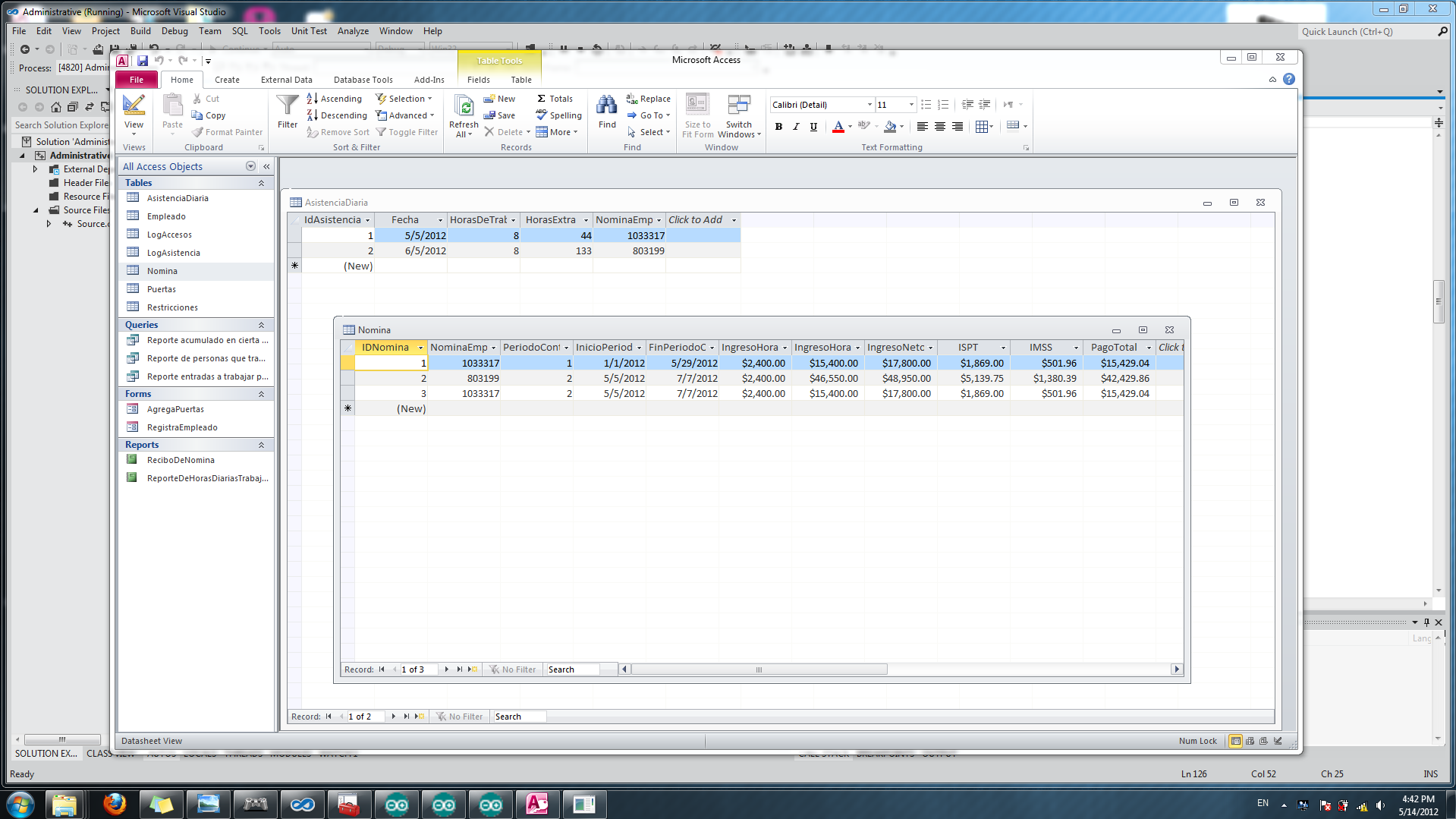
****

# Anexos

## Escenarios

* + - 1. Entrar a las instalaciones. Se verifica que no exista ningún registro anterior de entrada, que no tenga fecha y hora de salida, siendo así.
         1. El id de empleado de la tarjeta leida no esta registrado como empleado, se niega la solicitud de entrada y se registra como intento no permitido.
         2. El id de empleado esta registrado, tiene acceso a la hora que intenta entrar a las instalaciones, pero no tiene registrado acceso a la puerta, se niega la solicitud de entrada y se registra como intento no permitido.
         3. El id de empleado esta registrado, tiene acceso a la puerta, pero no esta en un horario disponible para ingresar, se niega la solicitud de entrada y se registra como intento no permitido.
         4. El id de empleado esta registrado, pero no tiene acceso a la puerta, ni esta en un horario permitido para ingresar, se niega la solicitud de entrada y se registra como intento no permitido.
         5. El id de empleado esta registrado, tiene acceso a la puerta, tiene permiso para entrar a la hora y día del intento, se permite la entrada a las instalaciones y se registra como entrada con la fecha y hora de entrada, dejando pendiente la fecha y hora de salida.
      2. Salir de las instalaciones. Se verifica que exista un registro de entrada, al cual le falte la fecha y hora de salida.
         1. El id de empleado es existente, se permite la salida de las instalaciones, se registra la fecha y hora de salida y se calculan las horas trabajadas restando de la hora de salida, la hora de entrada.

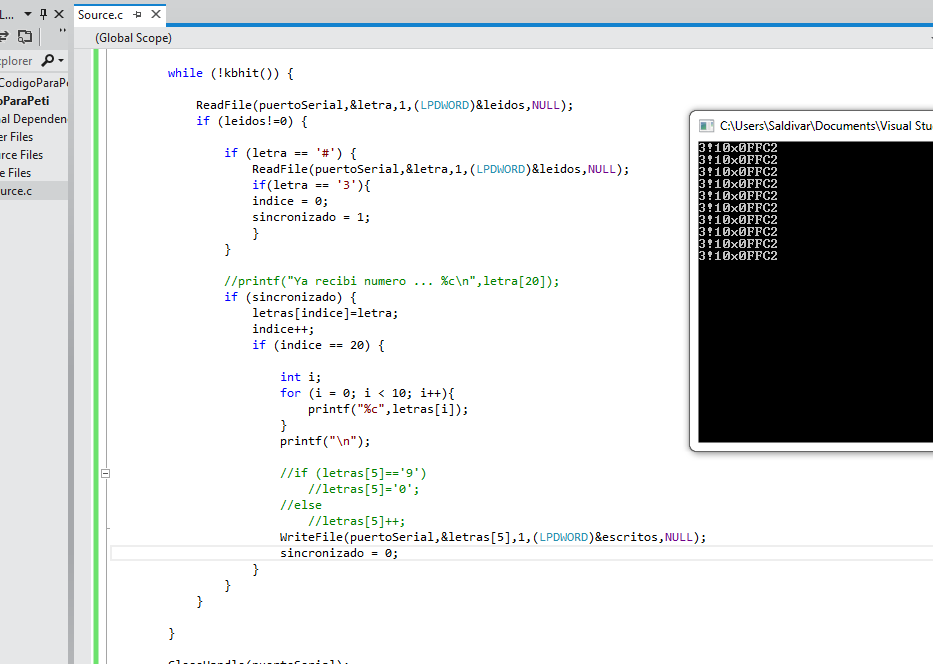
## Instancia del modelo relacional

****

## Pruebas unitarias y de integración

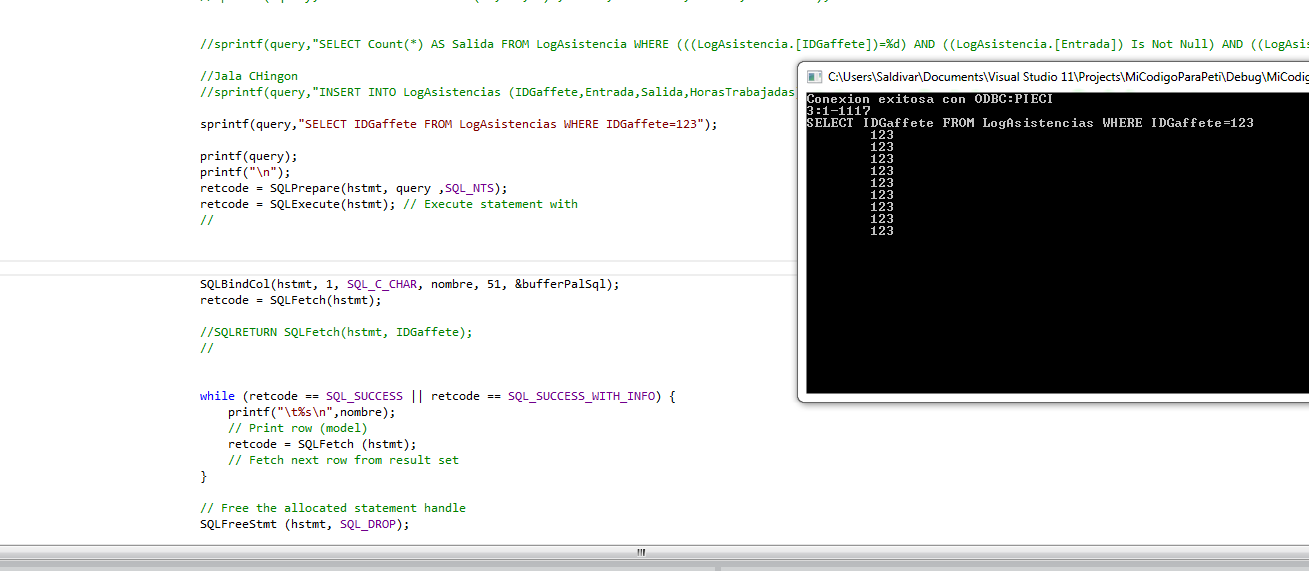
* + - 1. Pruebas para la sincronización de la comunicación entre los arduinos(XBee) y la aplicación.

Para sincronizar la comunicación se utilizaron dos bytes, así pues para empezar a leer los bytes de los datos de la tarjeta, primero debe recibir la aplicación un ‘#’, un ‘3’ y un ‘!’. A continuación se muestra como se imprimía en pantalla los bytes que recibía la aplicación, con lo que nos aseguramos que nuestra comunicación estaba perfectamente sincronizada.



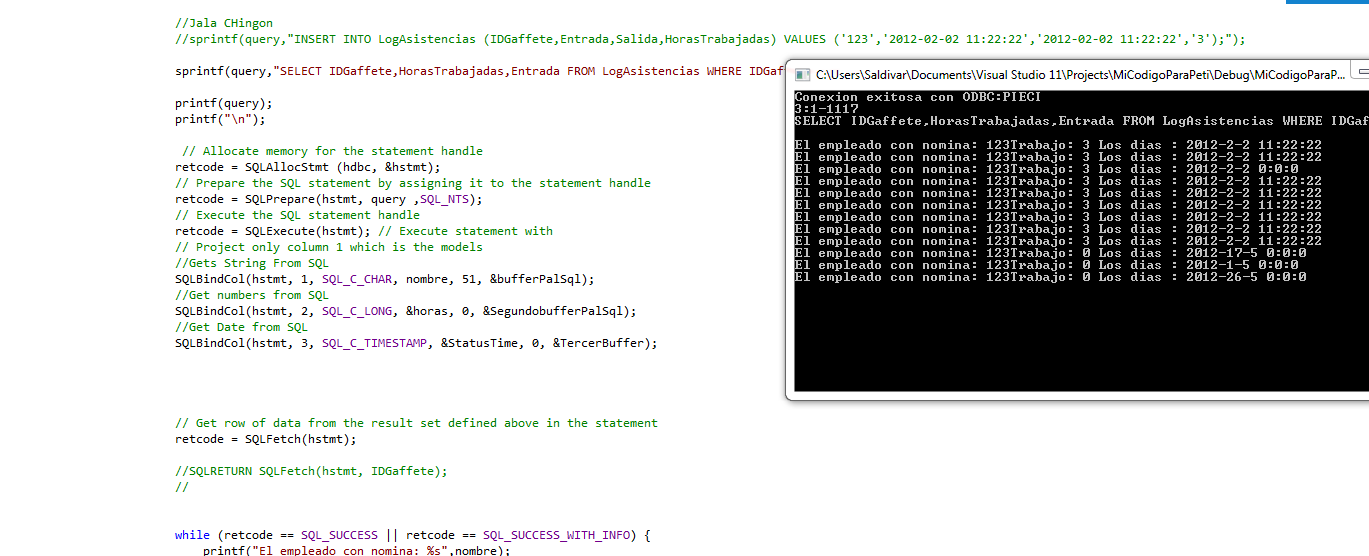
* + - 1. Extracción de datos de tablas de la base de datos.

A continuación se verificó que nos fue posible tomar datos de una tabla de la base de datos, al imprimir en pantalla los registros de la tabla LogAsistencias, donde el IDGaffete era igual a 123.



* + - 1. Pruebas de cambio de formato de fecha para su inserción en la base de datos.

A continuación se muestra el nuevo formato de la fecha:

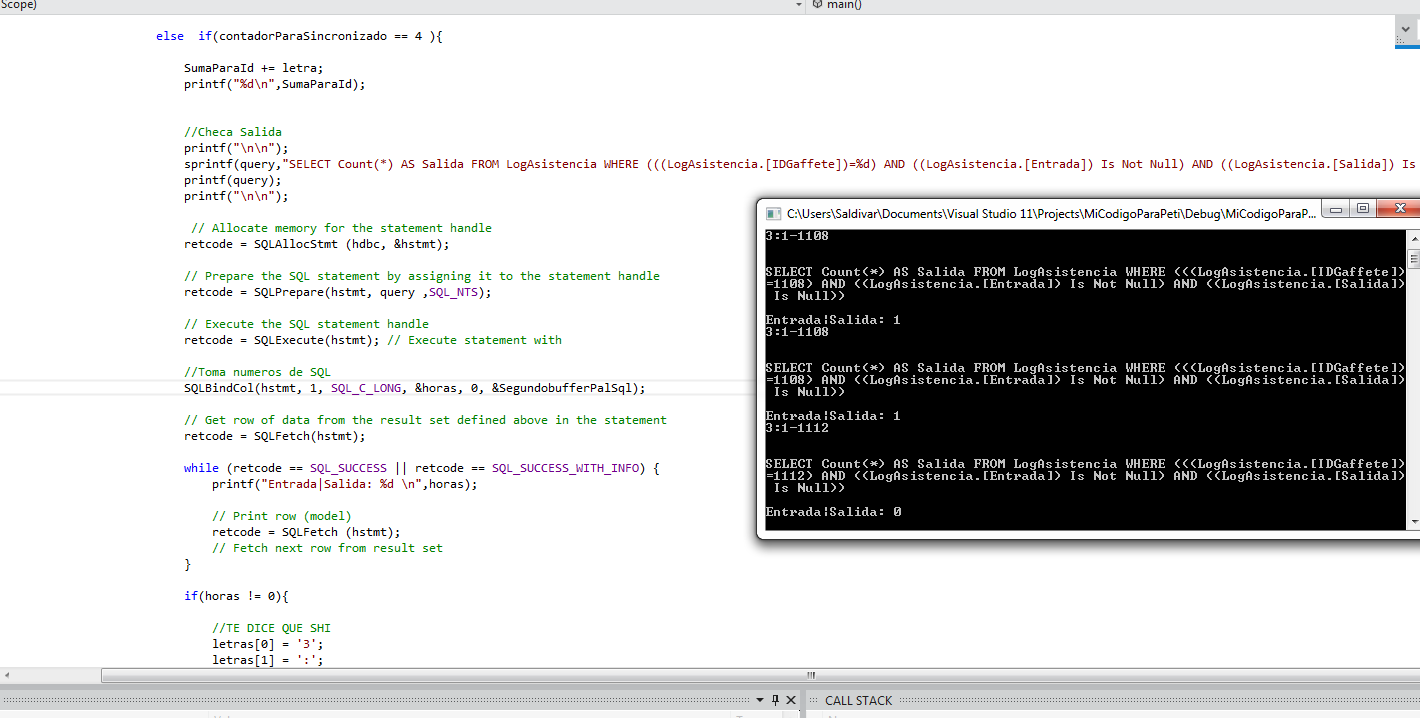


* + - 1. Prueba para detectar si el sistema puede definir si es una entrada, una salida o se rechaza la solicitud de entrada y dependiendo, si es una entrada, crea un nuevo registro en la tabla LogAsistencias, si es una salida, llena el campo de fecha y hora de salida del registro correspondiente en la tabla LogAsistencias.

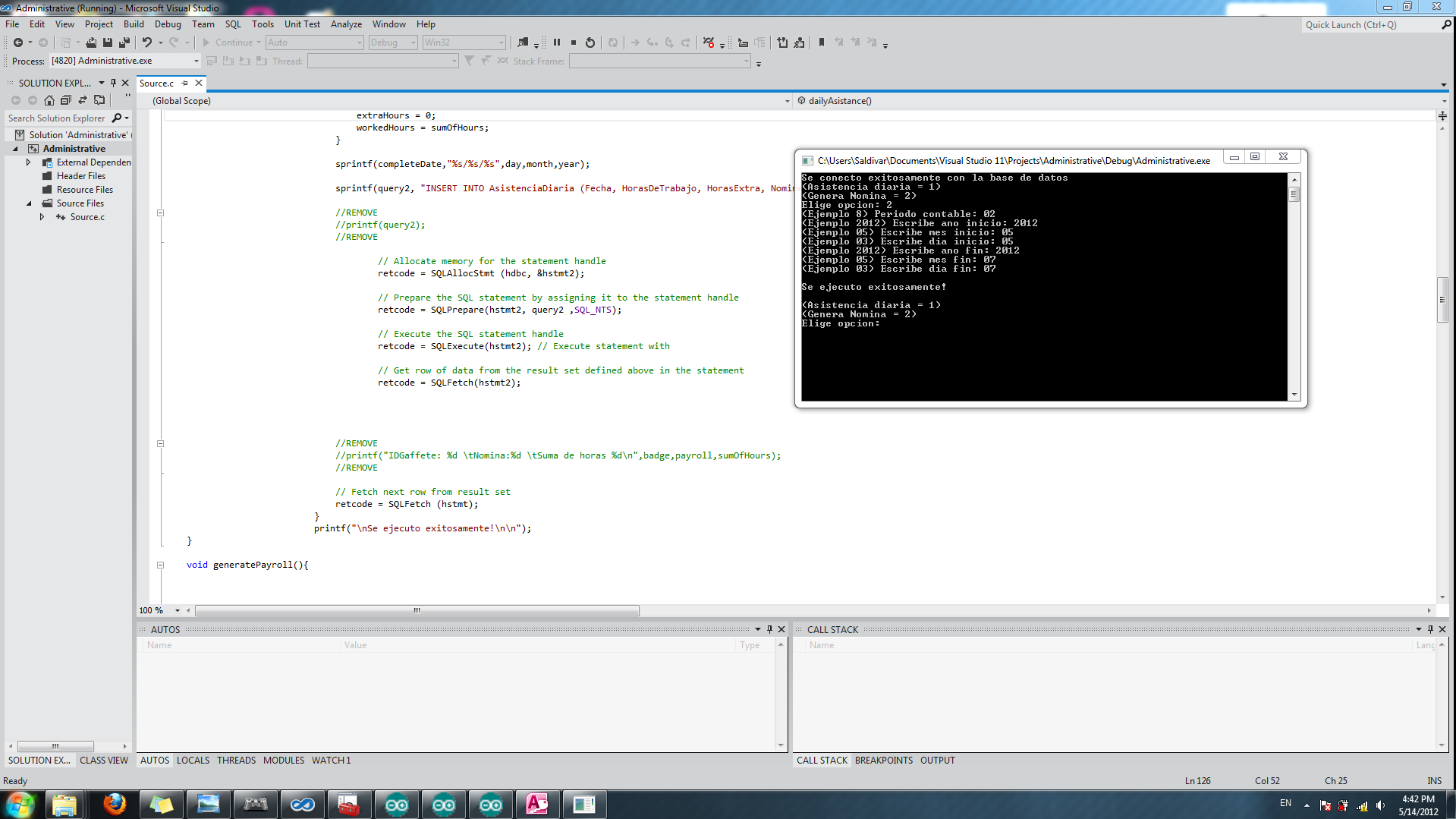
Si es una entrada imprime: entradasalida=0.

Si es una salida imprime: entradasalida=1.

Si no es permitida la entrada imprime: entradasalida=2.

****

7. Crea el pago de nómina de acuerdo a las horas trabajadas por periodo contable.

****

## Codigo de la aplicación impreso y documentado

### Código Lenguaje C-Aplicación

1 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* vv DOCUMENTING AND HELPFUL STUFF vv \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 2 //C\_Program\_Register\_SerialData\_AccessDB.c  
 3   
 4 //Compiled using Microsoft Visual Studio 11  
 5 //http://www.microsoft.com/visualstudio/en-us/products/2010-editions/express  
 6   
 7 //Created by Jorge Saldivar on May 13 2012.  
 8 //using Luis Ricardo Salgado example for academic purposes  
 9   
 10 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* vv DOCUMENTING AND HELPFUL STUFF vv \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 11   
 12 //Libraries used  
 13 #include <windows.h>  
 14 #include <conio.h>  
 15 #include <stdio.h>  
 16 #include <sql.h>  
 17 #include <sqlext.h>  
 18 #include <time.h>  
 19   
 20   
 21   
 22 //Variables used for the database  
 23 HENV henv; //Reference ("handle") to memory for environment variables / / ("Handle to environment")  
 24 HDBC hdbc; //Reference ("handle") to data connection (session) with the database ("Handle Conection to DataBase")  
 25 RETCODE retcode; //Return code for SQL operations ... actually 32-bit integer.  
 26 HSTMT hstmt; //Reference ("handle") to a statute ... actually is a STRING  
 27   
 28 //It keeps all the query in this chain of chars  
 29 char query[300];  
 30   
 31 //Variables for Serial port interaction  
 32 HANDLE serialPort;  
 33 DCB protocol;  
 34   
 35 //Chars to handle transfer of Serial Port  
 36 char oneLetter;  
 37 char multipleLetters[30];  
 38 int readFromSerial,writedToSerial;  
 39   
 40 //Variables for sync the serial data  
 41 int sync=0; //Sync data  
 42 int syncronizedForTwoPoints = 0; //Sync for protocol used  
 43 int counterForSync = 0; //Sincroniza los datos  
 44 int addForId; //Variable para guardar el id del usuario  
 45   
 46 //Variables for methods  
 47 int checkTheExit,checkTheTime,dayOfWeek;  
 48   
 49 //Variables that uses SQL to get data from tables  
 50 char door; //door number  
 51 char name[51]; //array of 51 chars  
 52 int hours,theDate,countDoors,validateTime;  
 53 SDWORD bufferForSql,secondBufferForSql,fourthBufferForSql,fifthBufferForSql;  
 54 SQLINTEGER thirdBufferForSql;  
 55 SQL\_TIMESTAMP\_STRUCT StatusTime; //Get time from SQL  
 56   
 57 //Funcion para checar la salida  
 58 int checkExit(int addForId, int door){  
 59   
 60 //REMOVE - it only prints data  
 61 printf("\n\n");  
 62 //REMOVE  
 63   
 64 //If the count is more than cero, it means the employee is inside the building  
 65 sprintf(query,"SELECT Count(\*) AS Salida FROM LogAsistencia WHERE (((LogAsistencia.[IDGaffete])=%d) AND ((LogAsistencia.[Entrada]) Is Not Null) AND ((LogAsistencia.[Salida]) Is Null))", addForId);  
 66   
 67 //REMOVE - it only prints data  
 68 printf(query);  
 69 printf("\n\n");  
 70 //REMOVE  
 71   
 72 // Allocate memory for the statement handle  
 73 retcode = SQLAllocStmt (hdbc, &hstmt);  
 74   
 75 // Prepare the SQL statement by assigning it to the statement handle  
 76 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
 77   
 78 // Execute the SQL statement handle  
 79 retcode = SQLExecute(hstmt); // Execute statement with  
 80   
81 // Takes SQL numbers  
 82 SQLBindCol(hstmt, 1, SQL\_C\_LONG, &hours, 0, &secondBufferForSql);  
 83   
 84 // Get row of data from the result set defined above in the statement  
 85 retcode = SQLFetch(hstmt);  
 86   
 87 while (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO) {   
 88 //Prints the exit hour  
 89 printf("Salida es 1 Entrada es 0 \nEntrada/Salida: %d \n",hours);  
 90   
 91 // Fetch next row from result set   
 92 retcode = SQLFetch (hstmt);  
 93 }  
 94   
 95 //If the count was more than cero, it means the employee tries to exit. Enter this function if true.  
 96 if(hours != 0){  
 97   
 98 //Protocol to send the message to open door  
 99 multipleLetters[0] = '3';  
100 multipleLetters[1] = ':';  
101 multipleLetters[2] = door;  
102 multipleLetters[3] = '-';  
103 multipleLetters[4] = 's';  
104 multipleLetters[5] = '\n';  
105 WriteFile(serialPort,&multipleLetters,6,(LPDWORD)&writedToSerial,NULL); //Sends the message  
106   
107 //REMOVE  
108 printf("\n\n");  
109 //REMOVE  
110   
111 //Registers the user in the access log and puts a 3 in autorizacion. 3 means exit  
112 sprintf(query,"INSERT INTO LogAccesos (IDGaffete, Puerta,Hora,Autorizado) VALUES ('%d','%d',NOW(),'3');",addForId,door-48);  
113   
114 //REMOVE  
115 printf(query);  
116 printf("\n\n");  
117 //REMOVE  
118   
119 // Allocate memory for the statement handle  
120 retcode = SQLAllocStmt (hdbc, &hstmt);  
121   
122 // Prepare the SQL statement by assigning it to the statement handle  
123 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
124   
125 // Execute the SQL statement handle  
126 retcode = SQLExecute(hstmt); // Execute statement with  
127   
128 //REMOVE  
129 printf("\n\n");  
130 //REMOVE  
131   
132 //Set the time when the employee exit the building  
133 sprintf(query,"UPDATE LogAsistencia SET Salida=NOW() WHERE LogAsistencia.IDGaffete = %d ;" ,addForId);  
134   
135 //REMOVE  
136 printf(query);  
137 printf("\n\n");  
138 //REMOVE  
139   
140 // Allocate memory for the statement handle  
141 retcode = SQLAllocStmt (hdbc, &hstmt);  
142   
143 // Prepare the SQL statement by assigning it to the statement handle  
144 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
145   
146 // Execute the SQL statement handle  
147 retcode = SQLExecute(hstmt); // Execute statement with  
148   
149 //REMOVE  
150 printf("\n\n");  
151 //REMOVE  
152   
153 //Set the hours work when the employee entered and get out of the building  
154 sprintf(query,"UPDATE LogAsistencia SET HorasTrabajadas=DateDiff('h',Entrada,NOW()) WHERE IDGaffete=%d;",addForId);  
155   
156 //REMOVE  
157 printf(query);  
158 printf("\n\n");  
159 //REMOVE  
160   
161 // Allocate memory for the statement handle  
162 retcode = SQLAllocStmt (hdbc, &hstmt);  
163   
164 // Prepare the SQL statement by assigning it to the statement handle  
165 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
166   
167 // Execute the SQL statement handle  
168 retcode = SQLExecute(hstmt); // Execute statement with  
169   
170 return 1;  
171 }  
172   
173 else {  
174 return 0;  
175 }  
176   
177 }  
178   
179   
  
181 int checkDoor(int addForId, int door){  
182   
183 //REMOVE  
184 printf("\n\n");  
185 //REMOVE  
186   
187 //Check if the user have access to the door. If count is more than 0 it means it does have access  
188 sprintf(query,"SELECT Count(\*) AS Puertas FROM Puertas WHERE IDGaffete=%d AND Puerta=%d",addForId,door-48);  
189   
190 //REMOVE  
191 printf(query);  
192 printf("\n\n");  
193 //REMOVE  
194   
195 // Allocate memory for the statement handle  
196 retcode = SQLAllocStmt (hdbc, &hstmt);  
197   
198 // Prepare the SQL statement by assigning it to the statement handle  
199 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
200   
201 // Execute the SQL statement handle  
202 retcode = SQLExecute(hstmt); // Execute statement with  
203   
204 // Takes SQL numbers  
205 SQLBindCol(hstmt, 1, SQL\_C\_LONG, &countDoors, 0, &fourthBufferForSql);  
206   
207 // Get row of data from the result set defined above in the statement  
208 retcode = SQLFetch(hstmt);  
209   
210 //If count is more than cero, it means it does have access. Enter the function if true.  
211 if(countDoors != 0){  
212 return 1;  
213   
214 }   
215   
216 else {  
217 return 0;  
218 }  
219   
220 }  
221   
222   
223 void checkTime(int addForId,int door){  
224   
225 //Day of week format   
226 /\*  
227 1 = Sunday  
228 2 = Monday  
229 3 = Tuesday  
230 4 = Wednesday  
231 5 = Thurday  
232 6 = Friday  
233 7 = Saturday  
234 \*/  
235   
236 //REMOVE  
237 printf("\n\n");  
238 //REMOVE  
239   
240 //Returns the day of week  
241 sprintf(query,"SELECT Weekday (NOW());");  
242   
243 //REMOVE  
244 printf(query);  
245 printf("\n\n");  
246 //REMOVE  
247   
248 // Allocate memory for the statement handle  
249 retcode = SQLAllocStmt (hdbc, &hstmt);  
250   
251 // Prepare the SQL statement by assigning it to the statement handle  
252 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
253   
254 // Execute the SQL statement handle  
255 retcode = SQLExecute(hstmt); // Execute statement with  
256   
257 // Takes SQL numbers  
258 SQLBindCol(hstmt, 1, SQL\_C\_LONG, &validateTime, 0, &fifthBufferForSql);  
259   
260 // Get row of data from the result set defined above in the statement  
261 retcode = SQLFetch(hstmt);  
262   
263 while (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO) {   
264 //Prints the exit hour  
265 printf("Domingo es igual a 1\nDia de la semana: %d \n",validateTime);  
266   
267 // Fetch next row from result set   
268 retcode = SQLFetch (hstmt);  
269 }  
270   
271 //If its Monday - Friday  
272 if(validateTime >= 2 && validateTime <= 6){  
273   
274   
275 //REMOVE  
276 printf("\n\n");  
277 //REMOVE  
278   
279 //Check if employee can enter between his access hours from Monday to friday  
280 sprintf(query,"SELECT Count(\*) As Valido FROM Restricciones WHERE time()>=Lu\_ViMin AND time()<=Lu\_ViMax AND IDGaffete=%d;",addForId);  
281   
282 //REMOVE  
283 printf(query);  
284 printf("\n\n");  
285 //REMOVE  
286   
287 // Allocate memory for the statement handle  
288 retcode = SQLAllocStmt (hdbc, &hstmt);  
289   
290 // Prepare the SQL statement by assigning it to the statement handle  
291 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
292   
293 // Execute the SQL statement handle  
294 retcode = SQLExecute(hstmt); // Execute statement with  
295   
296 // Takes SQL numbers  
297 SQLBindCol(hstmt, 1, SQL\_C\_LONG, &validateTime, 0, &fifthBufferForSql);  
298   
299 // Get row of data from the result set defined above in the statement  
300 retcode = SQLFetch(hstmt);  
301   
302 //Check if employee is on selected hours. Enter function  
303 if(validateTime != 0){  
304   
305   
306 //REMOVE  
307 printf("\n\n");  
308 //REMOVE  
309   
310 //Insert into access log that the employee has access to the door. 1 means autorization is valid.  
311 sprintf(query,"INSERT INTO LogAccesos (IDGaffete, Puerta,Hora,Autorizado) VALUES ('%d','%d',NOW(),'1');",addForId,door-48);  
312   
313 //REMOVE  
314 printf(query);  
315 printf("\n\n");  
316 //REMOVE  
317   
318 // Allocate memory for the statement handle  
319 retcode = SQLAllocStmt (hdbc, &hstmt);  
320   
321 // Prepare the SQL statement by assigning it to the statement handle  
322 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
323   
324 // Execute the SQL statement handle  
325 retcode = SQLExecute(hstmt); // Execute statement with  
326   
327   
328   
329 //REMOVE  
330 printf("\n\n");  
331 //REMOVE  
332   
333 //Insert the time the employee entered the building  
334 sprintf(query,"INSERT INTO LogAsistencia (IDGaffete, Entrada) VALUES ('%d',NOW());",addForId);  
335   
336 //REMOVE  
337 printf(query);  
338 printf("\n\n");  
339 //REMOVE  
340   
341 // Allocate memory for the statement handle  
342 retcode = SQLAllocStmt (hdbc, &hstmt);  
343   
344 // Prepare the SQL statement by assigning it to the statement handle  
345 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
346   
347 // Execute the SQL statement handle  
348 retcode = SQLExecute(hstmt); // Execute statement with  
349 }   
350   
351 else {  
352   
353 //Protocol to send the message to close door  
354 multipleLetters[0] = '3';  
355 multipleLetters[1] = ':';  
356 multipleLetters[2] = door;  
357 multipleLetters[3] = '-';  
358 multipleLetters[4] = 'n';  
359 multipleLetters[5] = '\n';  
360 WriteFile(serialPort,&multipleLetters,6,(LPDWORD)&writedToSerial,NULL); //Sends the message  
361   
362 //REMOVE  
363 printf("\n\n");  
364 //REMOVE  
365   
366 //Register in the access log that the employee tried to get in, but was not autorized. 2 is for not autorized.  
367 sprintf(query,"INSERT INTO LogAccesos (IDGaffete, Puerta,Hora,Autorizado) VALUES ('%d','%d',NOW(),'2');",addForId,door-48);  
368   
369 //REMOVE  
370 printf(query);  
371 printf("\n\n");  
372 //REMOVE  
373   
374 // Allocate memory for the statement handle  
375 retcode = SQLAllocStmt (hdbc, &hstmt);  
376   
377 // Prepare the SQL statement by assigning it to the statement handle  
378 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
379   
380 // Execute the SQL statement handle  
381 retcode = SQLExecute(hstmt); // Execute statement with  
382   
383 }  
384   
385 }  
386   
387 //If its Sunday  
388 else if (validateTime == 1){  
389   
390   
391 //REMOVE  
392 printf("\n\n");  
393 //REMOVE  
394   
395 //Check if employee can enter between his access hours on Sunday  
396 sprintf(query,"SELECT Count(\*) As Valido FROM Restricciones WHERE time()>=DomingoMin AND time()<=DomingoMax AND IDGaffete=%d;",addForId);  
397   
398 //REMOVE  
399 printf(query);  
400 printf("\n\n");  
401 //REMOVE  
402   
403 // Allocate memory for the statement handle  
404 retcode = SQLAllocStmt (hdbc, &hstmt);  
405   
406 // Prepare the SQL statement by assigning it to the statement handle  
407 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
408   
409 // Execute the SQL statement handle  
410 retcode = SQLExecute(hstmt); // Execute statement with  
411   
412 // Takes SQL numbers  
413 SQLBindCol(hstmt, 1, SQL\_C\_LONG, &validateTime, 0, &fifthBufferForSql);  
414   
415 // Get row of data from the result set defined above in the statement  
416 retcode = SQLFetch(hstmt);  
417   
418 if(validateTime != 0){  
419   
420   
421 //REMOVE  
422 printf("\n\n");  
423 //REMOVE  
424   
425 //Insert into access log that the employee has access to the door. 1 means autorization is valid.  
426 sprintf(query,"INSERT INTO LogAccesos (IDGaffete, Puerta,Hora,Autorizado) VALUES ('%d','%d',NOW(),'1');",addForId,door-48);  
427   
428 //REMOVE  
429 printf(query);  
430 printf("\n\n");  
431 //REMOVE  
432   
433 // Allocate memory for the statement handle  
434 retcode = SQLAllocStmt (hdbc, &hstmt);  
435   
436 // Prepare the SQL statement by assigning it to the statement handle  
437 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
438   
439 // Execute the SQL statement handle  
440 retcode = SQLExecute(hstmt); // Execute statement with  
441   
442   
443   
444 //REMOVE  
445 printf("\n\n");  
446 //REMOVE  
447   
448 //Insert the time the employee entered the building  
449 sprintf(query,"INSERT INTO LogAsistencia (IDGaffete, Entrada) VALUES ('%d',NOW());",addForId);  
450   
451 //REMOVE  
452 printf(query);  
453 printf("\n\n");  
454 //REMOVE  
455   
456 // Allocate memory for the statement handle  
457 retcode = SQLAllocStmt (hdbc, &hstmt);  
458   
459 // Prepare the SQL statement by assigning it to the statement handle  
460 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
461   
462 // Execute the SQL statement handle  
463 retcode = SQLExecute(hstmt); // Execute statement with  
464 }   
465   
466 else {  
467   
468 //Protocol to send the message to close door  
469 multipleLetters[0] = '3';  
470 multipleLetters[1] = ':';  
471 multipleLetters[2] = door;  
472 multipleLetters[3] = '-';  
473 multipleLetters[4] = 'n';  
474 multipleLetters[5] = '\n';  
475 WriteFile(serialPort,&multipleLetters,6,(LPDWORD)&writedToSerial,NULL); //Sends the message  
476   
477 //REMOVE  
478 printf("\n\n");  
479 //REMOVE  
480   
481 //Register in the access log that the employee tried to get in, but was not autorized. 2 is for not autorized.  
482 sprintf(query,"INSERT INTO LogAccesos (IDGaffete, Puerta,Hora,Autorizado) VALUES ('%d','%d',NOW(),'2');",addForId,door-48);  
483   
484 //REMOVE  
485 printf(query);  
486 printf("\n\n");  
487 //REMOVE  
488   
489 // Allocate memory for the statement handle  
490 retcode = SQLAllocStmt (hdbc, &hstmt);  
491   
492 // Prepare the SQL statement by assigning it to the statement handle  
493 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
494   
495 // Execute the SQL statement handle  
496 retcode = SQLExecute(hstmt); // Execute statement with  
497   
498 }  
499   
500 }  
501   
502 //If its Saturday  
503 else if(validateTime == 7){  
504   
505   
506 //REMOVE  
507 printf("\n\n");  
508 //REMOVE  
509   
510 //Check if employee can enter between his access hours on Saturday  
511 sprintf(query,"SELECT Count(\*) As Valido FROM Restricciones WHERE time()>=SabadoMin AND time()<=SabadoMax AND IDGaffete=%d;",addForId);  
512   
513 //REMOVE  
514 printf(query);  
515 printf("\n\n");  
516 //REMOVE  
517   
518 // Allocate memory for the statement handle  
519 retcode = SQLAllocStmt (hdbc, &hstmt);  
520   
521 // Prepare the SQL statement by assigning it to the statement handle  
522 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
523   
524 // Execute the SQL statement handle  
525 retcode = SQLExecute(hstmt); // Execute statement with  
526   
527 // Takes SQL numbers  
528 SQLBindCol(hstmt, 1, SQL\_C\_LONG, &validateTime, 0, &fifthBufferForSql);  
529   
530 // Get row of data from the result set defined above in the statement  
531 retcode = SQLFetch(hstmt);  
532   
533 if(validateTime != 0){  
534   
535   
536 //REMOVE  
537 printf("\n\n");  
538 //REMOVE  
539   
540 //Insert into access log that the employee has access to the door. 1 means autorization is valid.  
541 sprintf(query,"INSERT INTO LogAccesos (IDGaffete, Puerta,Hora,Autorizado) VALUES ('%d','%d',NOW(),'1');",addForId,door-48);  
542   
543 //REMOVE  
544 printf(query);  
545 printf("\n\n");  
546 //REMOVE  
547   
548 // Allocate memory for the statement handle  
549 retcode = SQLAllocStmt (hdbc, &hstmt);  
550   
551 // Prepare the SQL statement by assigning it to the statement handle  
552 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
553   
554 // Execute the SQL statement handle  
555 retcode = SQLExecute(hstmt); // Execute statement with  
556   
557   
558   
559 //REMOVE  
560 printf("\n\n");  
561 //REMOVE  
562   
563 //Insert the time the employee entered the building  
564 sprintf(query,"INSERT INTO LogAsistencia (IDGaffete, Entrada) VALUES ('%d',NOW());",addForId);  
565   
566 //REMOVE  
567 printf(query);  
568 printf("\n\n");  
569 //REMOVE  
570   
571 // Allocate memory for the statement handle  
572 retcode = SQLAllocStmt (hdbc, &hstmt);  
573   
574 // Prepare the SQL statement by assigning it to the statement handle  
575 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
576   
577 // Execute the SQL statement handle  
578 retcode = SQLExecute(hstmt); // Execute statement with  
579 }   
580   
581 else {  
582   
583 //Protocol to send the message to close door  
584 multipleLetters[0] = '3';  
585 multipleLetters[1] = ':';  
586 multipleLetters[2] = door;  
587 multipleLetters[3] = '-';  
588 multipleLetters[4] = 'n';  
589 multipleLetters[5] = '\n';  
590 WriteFile(serialPort,&multipleLetters,6,(LPDWORD)&writedToSerial,NULL); //Sends the message  
591   
592 //REMOVE  
593 printf("\n\n");  
594 //REMOVE  
595   
596 //Register in the access log that the employee tried to get in, but was not autorized. 2 is for not autorized.  
597 sprintf(query,"INSERT INTO LogAccesos (IDGaffete, Puerta,Hora,Autorizado) VALUES ('%d','%d',NOW(),'2');",addForId,door-48);  
598   
599 //REMOVE  
600 printf(query);  
601 printf("\n\n");  
602 //REMOVE  
603   
604 // Allocate memory for the statement handle  
605 retcode = SQLAllocStmt (hdbc, &hstmt);  
606   
607 // Prepare the SQL statement by assigning it to the statement handle  
608 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
609   
610 // Execute the SQL statement handle  
611 retcode = SQLExecute(hstmt); // Execute statement with  
612   
613 }  
614   
615 }  
616   
617   
618 }  
619   
620   
621 void main() {  
622   
623 serialPort = CreateFile("COM3",GENERIC\_READ|GENERIC\_WRITE,0,NULL,OPEN\_EXISTING,FILE\_ATTRIBUTE\_NORMAL,NULL); //opens the communication port for READ and WRITE  
624   
625 //If it can conect to the serial port, it enter t  
626 if (serialPort != INVALID\_HANDLE\_VALUE) {  
627   
628 //Initialize the serial protocol  
629 GetCommState(serialPort,&protocol);  
630 protocol.BaudRate = CBR\_9600;  
631 protocol.fBinary = TRUE;  
632 protocol.fParity = FALSE;  
633 protocol.ByteSize = 8;  
634 protocol.Parity = NOPARITY;  
635 protocol.StopBits = ONESTOPBIT;  
636 SetCommState(serialPort,&protocol);  
637   
638   
639 retcode = SQLAllocEnv(&henv); //Reference ("handle") memory space environment variables  
640 if (retcode == SQL\_SUCCESS) {  
641 retcode = SQLAllocConnect(henv, &hdbc); //Reference to session data  
642 if (retcode == SQL\_SUCCESS) {  
643 retcode = SQLConnect(hdbc, "dbPieci", SQL\_NTS, NULL, 0, NULL, 0); //Connect to database with ODBC  
644 if (retcode == SQL\_SUCCESS) {  
645 printf("Se conecto exitosamente con la base de datos\n");  
646 retcode = SQLAllocStmt(hdbc, &hstmt); //Reference to the SQL statement  
647 if (retcode == SQL\_SUCCESS) {  
648   
649 //The program is always running  
650 while(0 == 0) {  
651   
652 ReadFile(serialPort,&oneLetter,1,(LPDWORD)&readFromSerial,NULL); //Serial data read and stored in the variable char: letter  
653   
654 //If it read something it enters the function  
655 if (readFromSerial!=0) {  
656   
657 //Protocol to receive data by initializing the number 3  
658 if (oneLetter == '3') {  
659 syncronizedForTwoPoints = 1;  
660 printf("%c",oneLetter); //Prints number in screen  
661 }  
662   
663 //Is protocol "3:" is read it enters this function  
664 if (sync) {  
665   
666 if(counterForSync == 0 ){  
667 printf("%c",oneLetter); //Prints in screen  
668 door = oneLetter; //Saves door number  
669 }  
670   
671 else if(counterForSync == 1 ){  
672 printf("%c",oneLetter); //Prints in screen  
673 }  
674   
675 else if(counterForSync == 2 ){  
676 addForId = oneLetter \* 256; //Saves user ID part of it  
677 }  
678   
679 //Not do anything because of protocol char in 3 position doesnt have anything useful  
680 else if(counterForSync == 3 ){   
681 }  
682   
683 else if(counterForSync == 4 ){  
684   
 addForId += oneLetter; //Saves user ID correctly  
686 printf("%d\n",addForId); //Prints user ID  
687   
688 //CheckExit return 1 if its an exit  
689 checkTheExit = checkExit(addForId,door);  
690   
691 //If its not an exit, validates door.  
692 if (checkTheExit == 0){  
693 checkTheTime = checkDoor(addForId,door);  
694   
695 //If the employee is registered in the door, then check the time to know if the employee can enter  
696 if(checkTheTime == 1){  
697 checkTime(addForId,door);  
698 }  
699   
700 else {  
701   
702 //Protocol to send the message to close door  
703 multipleLetters[0] = '3';  
704 multipleLetters[1] = ':';  
705 multipleLetters[2] = door;  
706 multipleLetters[3] = '-';  
707 multipleLetters[4] = 'n';  
708 multipleLetters[5] = '\n';  
709 WriteFile(serialPort,&multipleLetters,6,(LPDWORD)&writedToSerial,NULL); //Sends the message  
710   
711 }  
712   
713 }  
714   
715   
716   
717 /\*  
718 //Get SQL Data con given Format  
719 //Get string  
720 SQLBindCol(hstmt, 1, SQL\_C\_CHAR, name, 51, &bufferForSql);  
721 //Get Numbers  
722 SQLBindCol(hstmt, 2, SQL\_C\_LONG, &hours, 0, &secondBufferForSql);  
723 //Get Date  
724 SQLBindCol(hstmt, 3, SQL\_C\_TIMESTAMP, &StatusTime, 0, &thirdBufferForSql);   
725 //Date is dislayed this way:  
726 printf("Los dias : %d-%d-%d %d:%d:%d\n",StatusTime.year,StatusTime.day,StatusTime.month,StatusTime.hour,StatusTime.minute,StatusTime.second);  
727 \*/  
728   
729 }   
730   
731 //Add Sync Counter  
732 counterForSync++;  
733   
734 //Resets the syncF  
735 if(counterForSync == 5 ){  
736 sync = 0;  
737 counterForSync = 0;  
738 }  
739   
740   
741 }  
742   
743 //If the letter 3 was previosly read, it enters this function  
744 if(oneLetter == ':' && syncronizedForTwoPoints == 1){  
745 printf("%c",oneLetter); //Prints number in screen  
746 syncronizedForTwoPoints = 0;  
747 sync = 1; //Syncs  
748 }  
749   
750   
751 }  
752 }  
753 }  
754 }  
755 }  
756 }  
757 // Free the allocated statement handle  
758 SQLFreeStmt (hstmt, SQL\_DROP);  
759 CloseHandle(serialPort);  
760 }  
761 }

### Código Lenguaje C- Administrativo

1 //Libraries used  
 2 #include <windows.h>  
 3 #include <conio.h>  
 4 #include <stdio.h>  
 5 #include <sql.h>  
 6 #include <sqlext.h>  
 7 #include <time.h>  
 8   
 9 //Variables used for the database  
 10 HENV henv; //Reference ("handle") to memory for environment variables / / ("Handle to environment")  
 11 HDBC hdbc; //Reference ("handle") to data connection (session) with the database ("Handle Conection to DataBase")  
 12 RETCODE retcode; //Return code for SQL operations ... actually 32-bit integer.  
 13 HSTMT hstmt; //Reference ("handle") to a statute ... actually is a STRING  
 14 HSTMT hstmt2;  
 15   
 16 //Chars to handle transfer of Serial Port  
 17 char option;  
 18 char query[300];  
 19 char query2[300];  
 20 char optionUser[5];  
 21   
 22 //Read string from keyboard   
 23 char year[5];   
 24 char month[3];   
 25 char day[3];  
 26 char optionKeyboard[5];  
 27   
 28 //For payroll  
 29 char countablePeriod[5];  
 30 char initialYear[5];   
 31 char initialMonth[3];   
 32 char initialDay[3];  
 33   
 34 int workedHoursIncome,extraHoursIncome, netIncome;  
 35 float ISPT, IMSS, grossIncome;  
 36   
 37 char finalYear[5];   
 38 char finalMonth[3];   
 39 char finalDay[3];  
 40   
 41 int sumOfWorkedHoursPayroll,sumOfExtraHoursPayroll;  
 42   
 43 //For query  
 44 int workedHours;  
 45 int extraHours;  
 46 int wage,extraWage;  
 47 char completeDate[14];  
 48 char initialDate[14];  
 49 char finalDate[14];  
 50   
 51 //Not exit program variable  
 52 int cycle = 1;  
 53   
 54 //Variables for methods  
 55 int checkTheExit,checkTheTime,dayOfWeek;  
 56   
 57 //Variables that uses SQL to get data from tables  
 58 char door; //door number  
 59 char name[51]; //array of 51 chars  
 60 int hours,sumOfHours,payroll,badge;  
 61 SDWORD bufferForSql,secondBufferForSql,fourthBufferForSql,fifthBufferForSql;  
 62 //SQLINTEGER thirdBufferForSql;  
 63 //SQL\_TIMESTAMP\_STRUCT StatusTime; //Get time from SQL  
 64   
 65   
 66 void dailyAsistance(){  
 67   
 68 printf( "(Ejemplo 2012) Escribe ano: " );  
 69   
 70 /\* notice stdin being passed in \*/  
 71 fgets ( year, 5, stdin );   
 72   
 73 //printf( "You entered a very long string, %s\n", year );  
 74   
 75 getchar();   
 76   
 77 printf( "(Ejemplo 05) Escribe mes: " );  
 78   
 79 /\* notice stdin being passed in \*/  
 80 fgets ( month, 3, stdin );   
 81   
 82 //printf( "You entered a very long string, %s\n", month );  
 83   
 84 getchar();   
 85   
 86 printf( "(Ejemplo 03) Escribe dia: " );  
 87   
 88   
 89 fgets ( day, 3, stdin );   
 90   
 91 //printf( "You entered a very long string, %s %s\n", day,month );  
 92   
 93 getchar();  
 94   
 95   
 96   
 97   
 98   
 99 sprintf(query, "SELECT E.IDGaffete, E.NominaEmpleado, Sum(LA.HorasTrabajadas) AS SumOfHorasTrabajadas FROM LogAsistencia AS LA, Empleado AS E WHERE (((E.IDGaffete)=[LA].[IDGaffete]) AND ((DatePart('yyyy',[Salida]))=%s) AND ((DatePart('m',[Salida]))=%s) AND ((DatePart('d',[Salida]))=%s)) GROUP BY E.IDGaffete, E.NominaEmpleado;", year,month,day);  
100   
101 // Allocate memory for the statement handle  
102 retcode = SQLAllocStmt (hdbc, &hstmt);  
103   
104 // Prepare the SQL statement by assigning it to the statement handle  
105 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
106   
107 // Execute the SQL statement handle  
108 retcode = SQLExecute(hstmt); // Execute statement with  
109   
110 // Takes SQL numbers  
111 SQLBindCol(hstmt, 1, SQL\_C\_LONG, &badge, 0, &fifthBufferForSql);  
112 SQLBindCol(hstmt, 2, SQL\_C\_LONG, &payroll, 0, &fifthBufferForSql);  
113 SQLBindCol(hstmt, 3, SQL\_C\_LONG, &sumOfHours, 0, &fifthBufferForSql);  
114   
115 // Get row of data from the result set defined above in the statement  
116 retcode = SQLFetch(hstmt);  
117   
118 while (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO) {   
119   
120 if(sumOfHours >= 8){  
121 workedHours = 8;  
122 extraHours = sumOfHours - workedHours;  
123 }  
124   
125 else{  
126 extraHours = 0;  
127 workedHours = sumOfHours;  
128 }  
129   
130 sprintf(completeDate,"%s/%s/%s",day,month,year);  
131   
132 sprintf(query2, "INSERT INTO AsistenciaDiaria (Fecha, HorasDeTrabajo, HorasExtra, NominaEmpleado) VALUES ('%s', %d, %d, %d)",completeDate,workedHours,extraHours,payroll);  
133   
134 //REMOVE  
135 //printf(query2);  
136 //REMOVE  
137   
138 // Allocate memory for the statement handle  
139 retcode = SQLAllocStmt (hdbc, &hstmt2);  
140   
141 // Prepare the SQL statement by assigning it to the statement handle  
142 retcode = SQLPrepare(hstmt2, query2 ,SQL\_NTS);  
143   
144 // Execute the SQL statement handle  
145 retcode = SQLExecute(hstmt2); // Execute statement with  
146   
147 // Get row of data from the result set defined above in the statement  
148 retcode = SQLFetch(hstmt2);  
149   
150   
151   
152   
153 //REMOVE  
154 //printf("IDGaffete: %d \tNomina:%d \tSuma de horas %d\n",badge,payroll,sumOfHours);  
155 //REMOVE   
156   
157 // Fetch next row from result set   
158 retcode = SQLFetch (hstmt);  
159 }  
160 printf("\nSe ejecuto exitosamente!\n\n");  
161 }  
162   
163 void generatePayroll(){  
164   
165   
166   
167 printf( "(Ejemplo 8) Periodo contable: " );  
168   
169 /\* notice stdin being passed in \*/  
170 fgets ( countablePeriod, 3, stdin );   
171   
 //printf( "You entered a very long string, %s\n", year );  
173   
174 getchar();   
175   
176   
177   
178 printf( "(Ejemplo 2012) Escribe ano inicio: " );  
179   
180 /\* notice stdin being passed in \*/  
181 fgets (initialYear, 5, stdin );   
182   
183 //printf( "You entered a very long string, %s\n", year );  
184   
185 getchar();   
186   
187   
188 printf( "(Ejemplo 05) Escribe mes inicio: " );  
189   
190 /\* notice stdin being passed in \*/  
191 fgets ( initialMonth, 3, stdin );   
192   
193 //printf( "You entered a very long string, %s\n", month );  
194   
195 getchar();   
196   
197 printf( "(Ejemplo 03) Escribe dia inicio: " );  
198   
199   
200 fgets ( initialDay, 3, stdin );   
201   
 //printf( "You entered a very long string, %s %s\n", day,month );  
203   
204 getchar();  
205   
206 printf( "(Ejemplo 2012) Escribe ano fin: " );  
207   
208 /\* notice stdin being passed in \*/  
209 fgets ( finalYear, 5, stdin );   
210   
211 //printf( "You entered a very long string, %s\n", year );  
212   
213 getchar();   
214   
215 printf( "(Ejemplo 05) Escribe mes fin: " );  
216   
217 /\* notice stdin being passed in \*/  
218 fgets ( finalMonth, 3, stdin );   
219   
220 //printf( "You entered a very long string, %s\n", month );  
221   
222 getchar();   
223   
224 printf( "(Ejemplo 03) Escribe dia fin: " );  
225   
226   
227 fgets ( finalDay, 3, stdin );   
228   
229 //printf( "You entered a very long string, %s %s\n", day,month );  
230   
231 getchar();  
232   
233   
234   
235 //printf("%s\t%s\t%s\t%s\t%s\t%s\t%s",countablePeriod,initialYear,initialMonth,initialDay,finalYear,finalMonth,finalDay);  
236   
237   
238   
239 sprintf(query, "SELECT SUM(HorasDeTrabajo) AS HorasDeTrabajoAcum, SUM(HorasExtra) AS HorasExtraAcum, AsistenciaDiaria.NominaEmpleado, Salario, SalarioExtra FROM AsistenciaDiaria, Empleado WHERE Empleado.NominaEmpleado = AsistenciaDiaria.NominaEmpleado AND ((DatePart('yyyy',[Fecha])) >= %s) AND ((DatePart('yyyy',[Fecha])) <= %s) AND ((DatePart('m',[Fecha])) >=%s) AND ((DatePart('m',[Fecha])) <=%s) AND ((DatePart('d',[Fecha])) >=%s) AND ((DatePart('d',[Fecha])) <=%s) GROUP BY AsistenciaDiaria.NominaEmpleado, Salario, SalarioExtra", initialYear,finalYear,initialMonth,finalMonth,initialDay,finalDay);  
240 //printf(query);   
241   
242 // Allocate memory for the statement handle  
243 retcode = SQLAllocStmt (hdbc, &hstmt);  
244   
245 // Prepare the SQL statement by assigning it to the statement handle  
246 retcode = SQLPrepare(hstmt, query ,SQL\_NTS);  
247   
248 // Execute the SQL statement handle  
249 retcode = SQLExecute(hstmt); // Execute statement with  
250   
251 // Takes SQL numbers  
252 SQLBindCol(hstmt, 1, SQL\_C\_LONG, &sumOfWorkedHoursPayroll, 0, &fifthBufferForSql);  
253 SQLBindCol(hstmt, 2, SQL\_C\_LONG, &sumOfExtraHoursPayroll, 0, &fifthBufferForSql);  
254 SQLBindCol(hstmt, 3, SQL\_C\_LONG, &badge, 0, &fifthBufferForSql);  
255 SQLBindCol(hstmt, 4, SQL\_C\_LONG, &wage, 0, &fifthBufferForSql);  
256 SQLBindCol(hstmt, 5, SQL\_C\_LONG, &extraWage, 0, &fifthBufferForSql);  
257   
258 // Get row of data from the result set defined above in the statement  
259 retcode = SQLFetch(hstmt);  
260   
261 while (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO) {   
262   
263 //printf("ENTRO");  
264 //printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\nSum of work = %d , sum of extra = %d , sum pf badge = %d wage = %d extra = %d\n\n",sumOfWorkedHoursPayroll,sumOfExtraHoursPayroll,badge,wage,extraWage);  
265   
266 sprintf(initialDate,"'%s/%s/%s'",initialMonth,initialDay,initialYear);  
267 sprintf(finalDate,"'%s/%s/%s'",finalMonth,finalDay,finalYear);  
268   
269 workedHoursIncome = wage \* sumOfWorkedHoursPayroll;   
270 extraHoursIncome = extraWage \* sumOfExtraHoursPayroll;   
271 netIncome = workedHoursIncome + extraHoursIncome;  
272 ISPT = (float) netIncome \* .105;  
273 IMSS = (float) netIncome \* .0282;  
274 grossIncome = (float) netIncome - ISPT - IMSS;  
275   
276 //printf("\n\nISTP: %.2f\t\tIMSS %.2f\t\tgrossIncome: %.2f", ISPT,IMSS,grossIncome);  
277   
278 sprintf(query2, "INSERT INTO Nomina (NominaEmpleado, PeriodoContable, InicioPeriodoContable, FinPeriodoContable, IngresoHorasDeTrabajo, IngresoHorasExtra, IngresoNeto, ISPT, IMSS, PagoTotal) VALUES (%d,%s,%s,%s,%d,%d,%d,%.2f,%.2f,%.2f)", badge, countablePeriod, initialDate, finalDate, workedHoursIncome, extraHoursIncome, netIncome, ISPT, IMSS, grossIncome);  
279 // printf(query2);  
280   
281   
282 // Allocate memory for the statement handle  
283 retcode = SQLAllocStmt (hdbc, &hstmt2);  
284   
285 // Prepare the SQL statement by assigning it to the statement handle  
286 retcode = SQLPrepare(hstmt2, query2 ,SQL\_NTS);  
287   
288 // Execute the SQL statement handle  
289 retcode = SQLExecute(hstmt2); // Execute statement with  
290   
291 // Get row of data from the result set defined above in the statement  
292 retcode = SQLFetch(hstmt2);  
293   
294   
295   
296   
297 // Fetch next row from result set   
298 retcode = SQLFetch (hstmt);  
299   
300 }  
301 printf("\nSe ejecuto exitosamente!\n\n");  
302   
303   
304 }  
305   
306   
307 void main() {  
308   
309   
310   
311   
312 retcode = SQLAllocEnv(&henv); //Reference ("handle") memory space environment variables  
313 if (retcode == SQL\_SUCCESS) {  
314 retcode = SQLAllocConnect(henv, &hdbc); //Reference to session data  
315 if (retcode == SQL\_SUCCESS) {  
316 retcode = SQLConnect(hdbc, "dbPieci", SQL\_NTS, NULL, 0, NULL, 0); //Connect to database with ODBC  
317 if (retcode == SQL\_SUCCESS) {  
318 printf("Se conecto exitosamente con la base de datos\n");  
319 retcode = SQLAllocStmt(hdbc, &hstmt); //Reference to the SQL statement  
320 if (retcode == SQL\_SUCCESS) {  
321   
322   
323 //cout << "Nomina" << endl;  
324   
325 //The program is always running  
326 while(cycle == 1) {  
327   
328 printf( "(Asistencia diaria = 1)\n(Genera Nomina = 2)\nElige opcion: " );  
329   
330 /\* notice stdin being passed in \*/  
331 fgets ( optionKeyboard, 5, stdin );   
332   
333 //printf( "You entered a very long string, %s\n", year );  
334   
335 //getchar();   
336   
337 sprintf(optionUser,"%s",optionKeyboard);  
338   
339   
340 if(optionUser[0] == 49){  
341 dailyAsistance();  
342 }  
343   
344 else if (optionUser[0] == 50){  
345 generatePayroll();  
346 }  
347   
348 else {  
349 printf("\nEhhh?\n");  
350 }  
351   
352   
353   
354   
355 }  
356   
357 }  
358 }  
359 }  
360 }  
361 }  
362

### Código Arduino – Conectado a lectoras RFID, Protocolo Wiegand

//IMPORTANT

//ARDUINO UNO MUST BE

//COMPILED USING A DLINE Mode and

//RUN USING UART Mode

//Libraries used

#include "pins\_arduino.h"

//Declaration for the PINS that the Arduino UNO used for the card reader wiegand

const int WiegandData1 = 4;

const int WiegandData0 = 5;

//Declaration for the PINS the ARDUINO UNO used to tell if a door opened or closed

const int ledVerde = 8;

const int ledRojo = 9;

const int bocina = 10;

//Wiegand Identifier

const String identificador = "3";

//Used for recieving a complete string from the serial communication

String inputString = "";

boolean stringComplete = false;

//Variables used for the wiegand so it can read the card bits

volatile long readerBits = 0;

volatile int readerBitCount = 0;

//Used for getting the user ID from the card.

//userIdB \* 256 + userIdA = userId as we know it

char userIdA;

char userIdB;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* vv Code for managing interruptions vv \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

All terminals can generate interrupts on ATmega168 transition.

The bit corresponding to the terminal to be the source of the event should

be enabled in the registry correspond PCInt and a service routine (ISR).

Because the registration PCInt operates port, not terminal, the routine ISR

must use some algorithm to implement an interrupt service routine

by terminal

Correspondenicas terminal and to interrupt masking registers:

D0-D7 => 16-23 = PCIR2 PCInt = PD = PCIE2 = pcmsk2

D8-D13 => 0-5 = PCIR0 PCInt = PB = PCIE0 = pcmsk0

A0-A5 (D14-D19) => 8-13 = PCIR1 PCInt = PC = PCIE1 = pcmsk1

\*/

volatile uint8\_t \*port\_to\_pcmask[] = {

&PCMSK0,

&PCMSK1,

&PCMSK2

};

typedef void (\*voidFuncPtr)(void);

volatile static voidFuncPtr PCintFunc[24] = {

NULL };

volatile static uint8\_t PCintLast[3];

void PCattachInterrupt(uint8\_t pin, void (\*userFunc)(void), int mode) {

uint8\_t bit = digitalPinToBitMask(pin);

uint8\_t port = digitalPinToPort(pin);

uint8\_t slot;

volatile uint8\_t \*pcmask;

if (mode != CHANGE) {

return;

}

// Modify the registry ("Interrupt Control Register") ICR

// As requested terminal, validating that is between 0 and 13

if (port == NOT\_A\_PORT) {

return;

}

else {

port -= 2;

pcmask = port\_to\_pcmask[port];

}

slot = port \* 8 + (pin % 8);

PCintFunc[slot] = userFunc;

// Set the interrupt mask

\*pcmask |= bit;

// Interrupt Enable

PCICR |= 0x01 << port;

}

static void PCint(uint8\_t port) {

uint8\_t bit;

uint8\_t curr;

uint8\_t mask;

uint8\_t pin;

// get the pin states for the indicated port.

curr = \*portInputRegister(port+2);

mask = curr ^ PCintLast[port];

PCintLast[port] = curr;

// mask is pins that have changed. screen out non pcint pins.

if ((mask &= \*port\_to\_pcmask[port]) == 0) {

return;

}

// mask is pcint pins that have changed.

for (uint8\_t i=0; i < 8; i++) {

bit = 0x01 << i;

if (bit & mask) {

pin = port \* 8 + i;

if (PCintFunc[pin] != NULL) {

PCintFunc[pin]();

}

}

}

}

SIGNAL(PCINT0\_vect) {

PCint(0);

}

SIGNAL(PCINT1\_vect) {

PCint(1);

}

SIGNAL(PCINT2\_vect) {

PCint(2);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ^^ Code for managing interruptions ^^ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\* vv Code for counting and storing bits vv \*\*\*\*\*\*\*\*\*\*

void readerOne(void) {

if(digitalRead(4) == LOW){

readerBitCount++;

readerBits = readerBits << 1; //Move the bits ...

readerBits |= 1; // ... add a bit to '1 'in the least significant bit

}

}

void readerZero(void) {

if(digitalRead(5) == LOW){

readerBitCount++;

readerBits = readerBits << 1; //Move the bits ...

}

}

//\*\*\*\*\*\*\*\*\*\*\* ^^ Code for counting and storing bits ^^ \*\*\*\*\*\*\*\*\*\*

void setup()

{

//Serial transfer speed

Serial.begin(9600);

// Attach pin change interrupt service routines from the Wiegand RFID readers

PCattachInterrupt(WiegandData1, readerOne, CHANGE);

PCattachInterrupt(WiegandData0, readerZero, CHANGE);

delay(10);

// the interrupt in the Atmel processor misses out the first negative pulse as the inputs are already high,

// so this gives a pulse to each reader input line to get the interrupts working properly.

// Then clear out the reader variables.

// The readers are open collector sitting normally at a one so this is OK

for(int i = WiegandData1; i<=WiegandData0; i++){

pinMode(i, OUTPUT);

digitalWrite(i, HIGH); // enable internal pull up causing a one

digitalWrite(i, LOW); // disable internal pull up causing zero and thus an interrupt

pinMode(i, INPUT);

digitalWrite(i, HIGH); // enable internal pull up

}

delay(10);

// put the reader input variables to zero

readerBits=0;

readerBitCount = 0;

// enable pins of the LEDs

pinMode(ledVerde, OUTPUT);

pinMode(ledRojo, OUTPUT);

digitalWrite(13, HIGH); // show Arduino has finished initilisation

}

void loop() {

//If the card is slided in the card reader enters the condition

if(readerBitCount >= 26){

//Protocol to write on serial

Serial.write("3");

Serial.write(":");

Serial.write("1");

Serial.write("-");

//Gets the user id

int userID = readerBits >> 1 & 0xffff;

int a,b;

a = userID & 0xff;

b = userID >> 8 & 0xff;

userIdA = (char)a;

userIdB = (char)b;

//Sends the user id in 1 Byte.

//Serial.write can only send 1 byte at a time

Serial.write(userIdB);

Serial.write("+");

Serial.write(userIdA);

readerBits = 0;

readerBitCount = 0;

}

//String complete received from serial communication

if(stringComplete){

//Protocol to turn on led on on green or red and making a sound or not

if(inputString[0] == '3' && inputString[1] == ':' && inputString[2] == '1' && inputString[3] == '-' && inputString[4] == 's'){

digitalWrite(ledVerde, HIGH);

tone(bocina, 2000);

delay(1000);

digitalWrite(ledVerde, LOW);

noTone(bocina);

} else if(inputString[0] == '3' && inputString[1] == ':' && inputString[2] == '1' && inputString[3] == '-' && inputString[4] == 'n'){

digitalWrite(ledRojo, HIGH);

delay(1000);

digitalWrite(ledRojo, LOW);

}

inputString=""; //inputString reset to null

stringComplete=false;

}

}

//Sets serial communication, always ready to receive

void serialEvent(){

while(Serial.available()){

char inChar = (char)Serial.read(); //read char from serial

inputString += inChar; //Creating string

if(inChar == '\n'){ //if string is an "enter" the string is complete

stringComplete = true;

}

}

}

### Código Arduino- conectado a base de datos.

/\*

Software serial multple serial test

Receives from the hardware serial, sends to software serial.

Receives from software serial, sends to hardware serial.

The circuit:

\* RX is digital pin 2 (connect to TX of other device)

\* TX is digital pin 3 (connect to RX of other device)

created back in the mists of time

by Tom Igoe

based on Mikal Hart's example

This example code is in the public domain.

\*/

#include <SoftwareSerial.h>

const int pinRX = 2;

const int pinTX = 3;

SoftwareSerial XBee(pinRX, pinTX); // RX, TX

void setup()

{

// Set data rates

Serial.begin(9600);

// set the data rate for the SoftwareSerial port

XBee.begin(9600);

}

void loop() // run over and over

{

if (XBee.available())

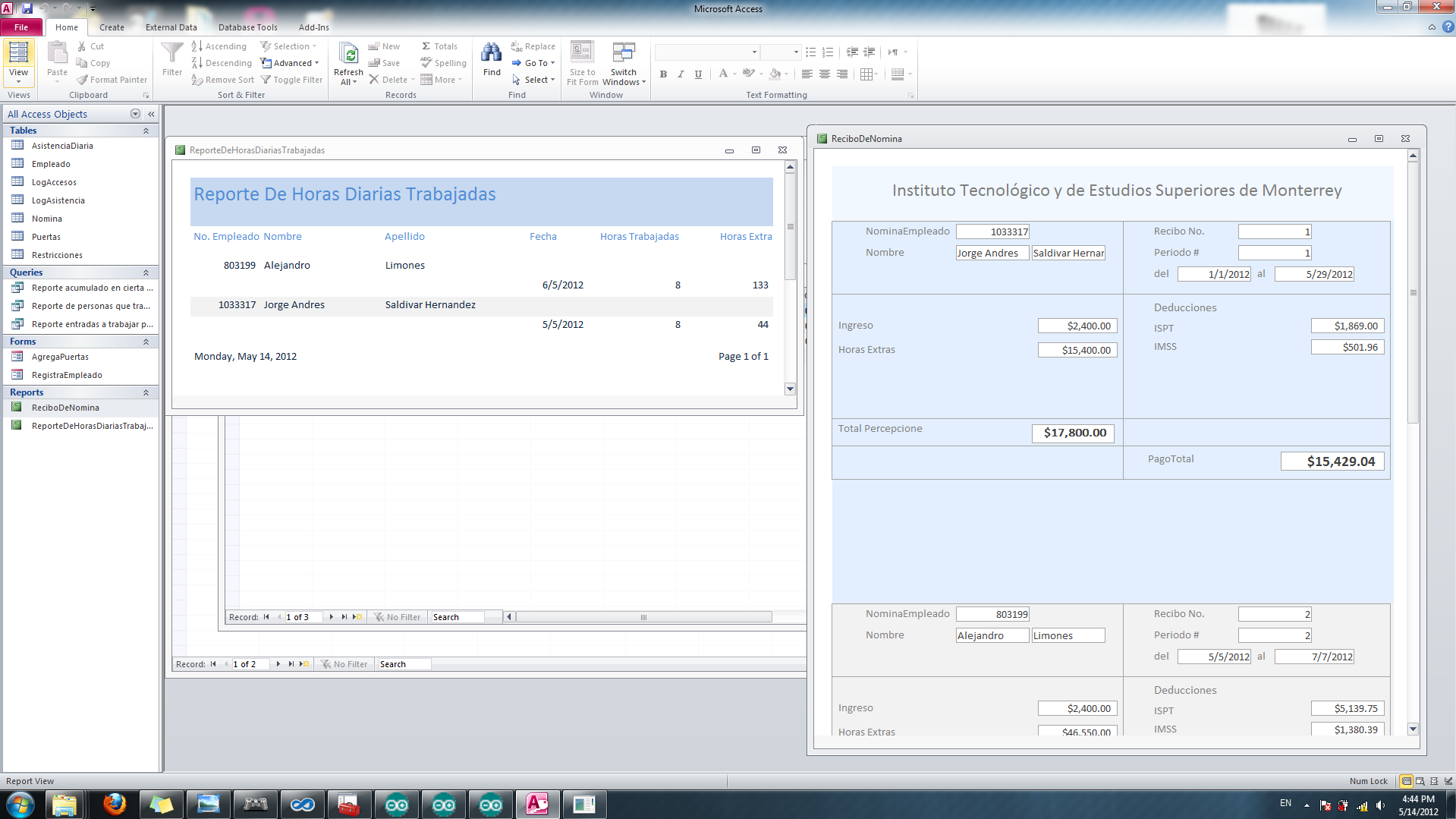
Serial.write(XBee.read());

if (Serial.available())

XBee.write(Serial.read());

}

## Ejemplos de todos los reportes con datos significativos

****