

INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

CIRCUITOS LÓGICOS

REGISTROS

Implementación de un Taxímetro de Tarifa Variable

170404

Silvestre GONZÁLEZ ABREU

173347

Andrea DE ANDA KURI

11 de noviembre, 2019

ITam

Introducción

En el presente proyecto se tuvo el objetivo de desarrollar un taxímetro capaz de calcular la tarifa de un viaje con base en más de ocho variables de entrada. El enfoque principal fue implementarlo de tal manera que fuera sencillo de instalar y desinstalar; no sólo por tratarse de una modificación temporal para los autos utilizados, pero también para crear un sistema modular que puede ser compartido por varios autos sin necesidad del uso de herramientas.

La mayor parte de las restricciones fueron planteadas como consecuencia de las limitaciones de memoria del sistema utilizado, limitaciones de los componentes disponibles o límites definidos para acotar la complejidad de la solución. Algunas de las restricciones fueron definidas en la fase de pruebas al descubrir limitaciones no previstas durante la etapa de planeación, éstas fueron analizadas para encontrar la fuente del problema, y una solución adecuada o, en su defecto, plantear una restricción justificada.

A lo largo de este proyecto fue necesario tener medidas de seguridad especiales, pues se levantaron los autos utilizados para poder instalar los sensores requeridos y se usaron herramientas como el cautín a altas temperaturas para realizar puntos de soldadura o herramientas de corte. A pesar de que hubo accidentes inesperados en el desarrollo del proyecto, sólo representaron retrasos al itinerario del mismo y riesgos graves para los participantes y colaboradores.

Por último, debido a la naturaleza del problema se requirió de un gran número de pruebas. Inicialmente se probaron de manera individual todos los componentes para descartarlos como posibles fuentes de error en pruebas subsecuentes. En la segunda etapa de pruebas se comprobó la comunicación de la tarjeta Arduino con cada uno de los sensores, displays, botones y módulos usados, nuevamente para descartar fuentes de error. En la tercera etapa se realizaron pruebas del dispositivo terminado pero no instalado en los vehículos, es decir, pruebas estáticas para comprobar el código realizado con excepción de la medición de distancia de viaje. Por último, se realizaron las pruebas dinámicas. Las primeras de ellas se hicieron en espacios controlados, como estacionamientos y calles poco transitadas a bajas velocidades; y finalmente se pasó a probar velocidades cercanas a los límites de velocidad en vías de acceso controlado.

En resumen, este proyecto consistió en desarrollar el taxímetro con componentes no especializados en el tema, es decir, sin adquirir piezas propias de un taxímetro existente y crear una nueva solución con los conocimientos que tenemos. La gran cantidad de variables de entrada crea una variedad de casos y estados a considerar, por lo que se deben encontrar maneras de agrupar y simplificar el código. Como ya se mencionó, otro factor de complejidad fue la necesidad de instalar de forma no permanente pero segura los sensores necesarios a los vehículos de prueba.

Planteamiento del problema

Se plantea el problema de la falta de un dispositivo de instalación sencilla y no permanente para llevar el conteo de la tarifa de un viaje en taxi con el uso de registros y memoria. La mayor parte de los taxímetros actualmente en uso requieren de softwares especializados o modificaciones de hardware para hacer cambios de tarifa y/o cambiar las variables del sistema. Como consecuencia, la solución propuesta es fácil de modificar, pues al estar basada en Arduino se tiene acceso a un software gratuito para alterar los parámetros de cálculo de tarifa. Además, es necesario considerar la posibilidad de añadir los datos de cálculo para más de 30 vehículos y seleccionar el auto en cuestión sin necesidad de usar la computadora.

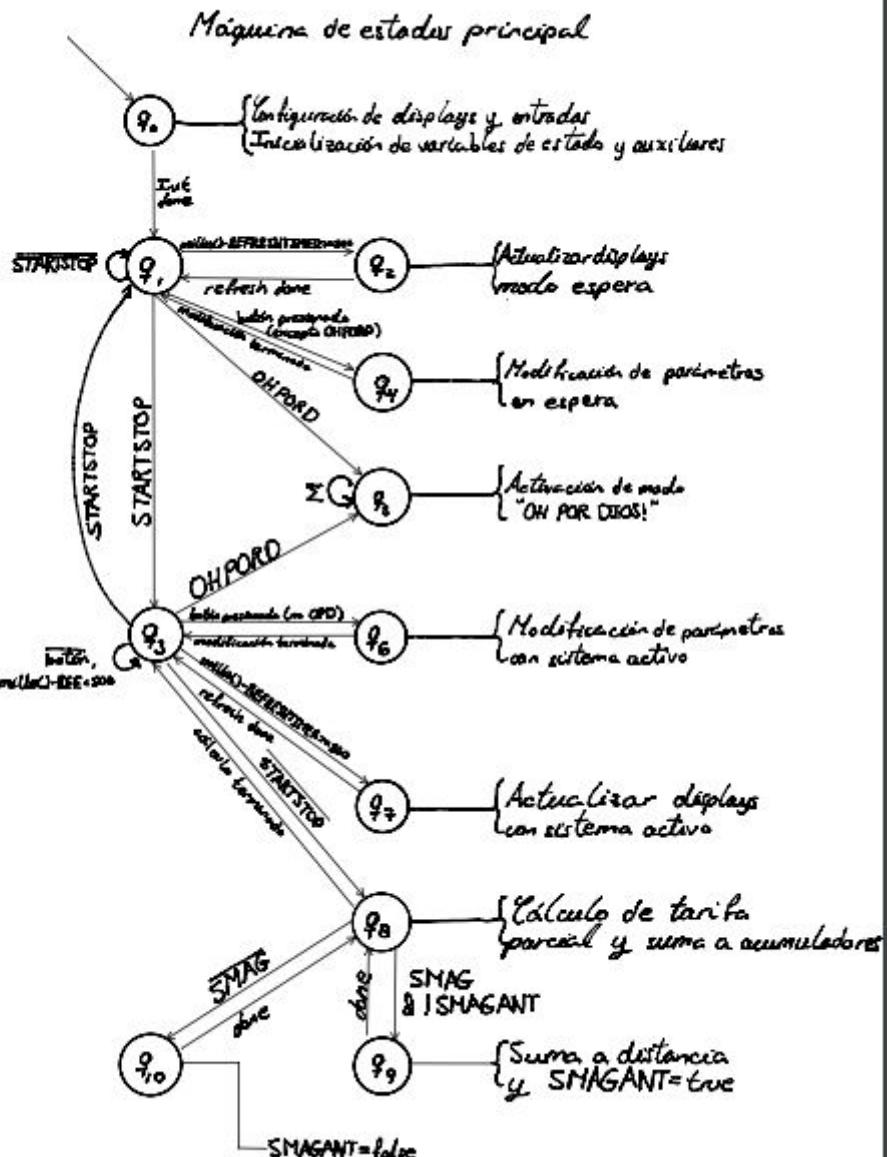
Además, se planteó un presupuesto máximo de MXN\$1,500 para la implementación de la solución generada, pues el precio mínimo de un taxímetro comercial se encuentra alrededor de MXN\$1,000. Se aumenta el presupuesto sobre el precio comercial, pues se considera que al tratarse de una solución con un mayor número de variables puede requerir una mayor inversión.

Restricciones

1. El taxímetro no se mantiene encendido por más de 45 días de forma continua, pues para almacenar el tiempo transcurrido desde que se encendió el dispositivo se utilizó una variable de tipo unsigned long. Ésta es una variable de 32 bits (4 bytes) y el tiempo es medido en milisegundos, lo que se traduce un máximo de 4,294,967,295 milisegundos de funcionamiento antes de un desbordo, es decir, 49.71 días.
2. A lo largo de un viaje pueden agregarse pasajeros (hasta 4), pero el total de pasajeros terminan sus viajes en un mismo punto. Esta restricción se tiene como consecuencia de la cantidad de puertos disponibles en la tarjeta Arduino, pues para permitir la finalización de viaje de forma individual sería necesario un botón adicional y la totalidad de las entradas digitales de la tarjeta se utilizan para la solución presentada.
3. No se implementa en el dispositivo la medición de distancia de viaje por medio de GPS debido al costo y a la menor precisión del mismo.
4. Los vehículos de prueba no deben circular a más de 55 km/h. (Este punto es discutido con mayor detalle en la sección de análisis de resultados)
5. Se asume que no se puede presionar más de un botón a la vez debido a la elevada velocidad de reloj de 16MHz, por lo que se registra sólo el primero en ser presionado.

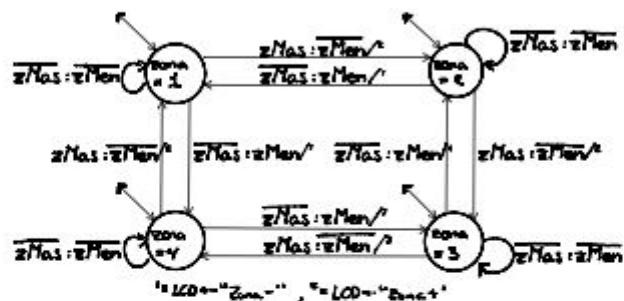
Máquinas de estado

Máquina de estados taxímetro



Maquinaria de estado secundarias

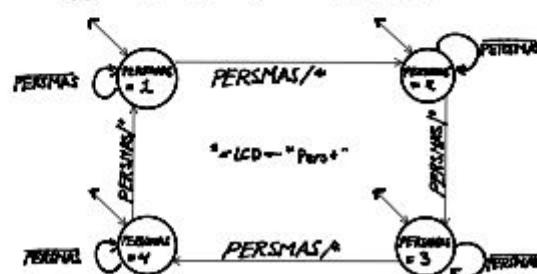
q4 si botón presionado es zMas ó zMen



q4 si botón presionado es COCHE

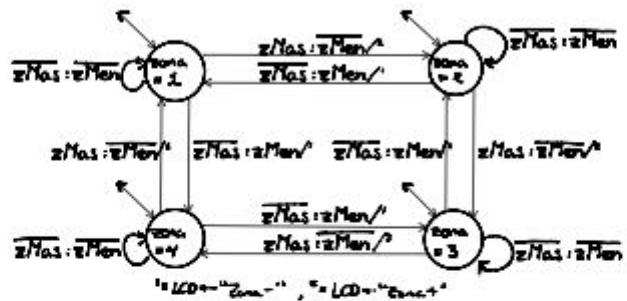


q4 si el botón presionado es PERSMAS

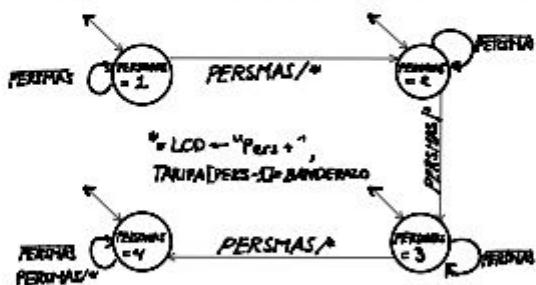


si el botón es otro, q_7 es una asignación

q_6 si botón presionado es $\bar{z}\text{Mas}$ o $\bar{z}\text{Men}$

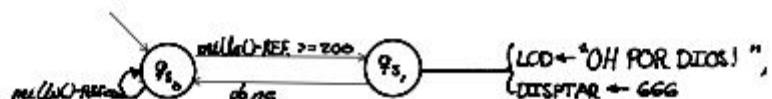


q_6 si el botón presionado es PERSMAS



si el botón es otro, q_6 es una asignación

q_5 :



Análisis de entradas y salidas

Tabla de entradas

| Symbol | Descripción | Tipo de variable |
|-----------|---|----------------------|
| STARTSTOP | Botón de activación y desactivación del sistema | Entrada (bit) |
| ZMAS | Botón para aumentar la zona de viaje | Entrada (bit) |
| ZMEN | Botón para reducir la zona de viaje | Entrada (bit) |
| COCHE | Botón para alternar el coche en uso entre los almacenados | Entrada (bit) |
| PERSMAS | Botón para aumentar en uno la variable de estado PERSONAS | Entrada (bit) |
| PARAMAS | Botón para aumentar el número de paradas | Entrada (bit) |
| DIAMAS | Botón para aumentar el día de la semana | Entrada (bit) |
| HORAMAS | Botón para aumentar 20 minutos a la hora actual del sistema | Entrada (bit) |
| HORAMEN | Botón para restar 20 minutos a la hora actual del sistema | Entrada (bit) |
| SMAG | Sensor magnético para detección de giro de las ruedas | Entrada (bit) |
| OHPORD | Botón para activar el modo OHPORDIOS | Entrada (bit) |
| DISPTAR | Número entero de 4 dígitos a mostrar en los 7 segmentos | Salida (int) |
| DISTANCIA | Cantidad de kilómetros recorridos | Salida (double) |
| ACTIVESTR | Parte del mensaje a desplegar en LCD | Salida (String) |
| TARIFA[4] | Arreglo para almacenar las tarifas por persona | Salida (double[4]) |
| LCD | Texto a desplegar en la pantalla LCD | Salida (String) |
| ACTIVE | Indica si el taxímetro está corriendo actualmente | Estado (bool) |
| ISCOCHE | Indica si el vehículo seleccionado es auto o camioneta | Estado (bool) |
| ZONA | Indica la zona seleccionada actual | Estado (int) |
| PERSONAS | Indica el número de personas seleccionado | Estado (int) |
| RTC | Módulo de fecha y hora actual | Estado (RtcDateTime) |
| MODODIOS | Indica si actualmente está activado el modo OHPORDIOS | Estado (bool) |

Tabla de salidas

| Acción | Evento | Significado entrada |
|--|---------------------|---|
| MODODIOS ← true • - | OHPORD | 1: se presionó el botón del modo 0: no se presionó |
| ACTIVE ← true ACTIVESTR ← “” PARADAS ← 0 PERSONAS ← 1 STARTTIME ← millis() VUELTAS ← 0 DISTANCIA ← 0 TARIFA[] ← 0 TARIFA[0] ← banderazo | STARTSTOP ACTIVE | 1: se presionó el botón para activar & false: no estaba activo |
| ACTIVE ← false ACTIVESTR ← “Inactivo” PERSONAS ← 1 | | 1: se presionó el botón para activar & true: estaba activo |
| • - | | 0: no se presionó |
| RTC ← RTC + 20min LCD ← “20min+” | HORAMAS ACTIVE | 1: se presionó el botón & false: en espera |
| LCD ← “Bloqueado” | | 1: se presionó el botón & true: activo |
| • - | | 0: no se presionó |
| RTC ← RTC - 20min LCD ← “20min-” | HORAMEN ACTIVE | 1: se presionó el botón & false: en espera |
| LCD ← “Bloqueado” | | 1: se presionó el botón & true: activo |
| • - | | 0: no se presionó |
| RTC ← RTC + 1d LCD ← “Dia+” | DIAMAS ACTIVE | 1: se presionó el botón & false: en espera |
| LCD ← “Bloqueado” | | 1: se presionó el botón & true: activo |
| • - | | 0: no se presionó |
| ZONA ← ZONA + 1 LCD ← “Zona+” | ZMAS ZONA | 1: se presionó el botón & <4: se tiene una zona 1, 2 o 3 |
| ZONA ← 1 LCD ← “Zona+” | | 1: se presionó el botón & 4: se tiene zona 4 |
| • - | | 0: no se presionó |

| | | |
|--|-------------------------------|--|
| ZONA ← ZONA - 1 LCD ← "Zona-" | ZMEN ZONA | 1: se presionó el botón & >1: se tiene una zona 4, 3 o 2 |
| ZONA ← 4 LCD ← "Zona-" | | 1: se presionó el botón & 1: se tiene zona 1 |
| • - | | 0: no se presionó |
| LCD ← "Bloqueado" | COCHE ACTIVE | 1: se presionó el botón & true: estaba activo el sistema |
| ISCOCHE ← !ISCOCHE LCD ← ISCOCHE | | 1: se presionó el botón & false: estaba en espera el sistema |
| • - | | 0: no se presionó |
| TARIFA[PERSONAS] ← BANDERAZO PERSONAS ← 1 LCD ← "Pers+" | PERSMAS PERSONAS ACTIVE | 1: se presionó el botón & 4: se tiene 4 personas & false: estaba en espera |
| TARIFA[PERSONAS] ← BANDERAZO PERSONAS ← PERSONAS + 1 LCD ← "Pers+" | | 1: se presionó el botón & <4: se tiene 1, 2 o 3 personas |
| • - | | 0: no se presionó |
| PARADAS ← PARADAS + 1 LCD ← "Para+" | PARAMAS ACTIVE | 1: se presionó el botón & true: estaba activo el sistema |
| LCD ← "Bloqueado" | | 1: se presionó el botón & false: estaba en espera el sistema |
| • - | | 0: no se presionó |
| TARIFAPAR ← 0 VUELTAS ← VUELTAS + 1 TARIFAPAR ← TARIFAPAR + PRECIOV TRAIFAPAR ← TARIFAPAR + PRECIOT | ACTIVE SMAG | true: sistema activo & 1: se registró vuelta de rueda |
| • - | | |
| TARIFA[0] ← TARIFA[0] + TARIFAPAR | | |
| • - | | |
| TARIFA[PERSONAS-1] ← TARIFA[PERSONAS-1] + TARIFAPAR | | |

| | | |
|---------------------------|--------------|---|
| TARIFAPAR ← 0 | | true: sistema activo |
| TRAIFAPAR ← TARIFAPAR + | | & 0: no se registró vuelta de rueda |
| PRECIOT | | |
| . | | |
| . | | |
| TARIFA[0] ← TARIFA[0] + | | |
| TARIFAPAR | | |
| . | | |
| . | | |
| TARIFA[PERSONAS-1] ← | | |
| TARIFA[PERSONAS-1] + | | |
| TARIFAPAR | | |
| . | | |
| • - | | false: sistema en espera |
| LCD ← “” | REFRESHTIMER | |
| LCD ← ACTIVESTR | MODODIOS | |
| LCD ← DISTANCIA + “km” | ACTIVE | millis()-refreshTimer>=500: han pasado más de 500 milisegundos desde el último refresco de displays & false: no modo OHPORDIOS & true: sistema activo |
| LCD ← (millis() - | | |
| STARTTIME)/60000 | | |
| DISPTAR ← TARIFA[0]*10 | | |
| LCD ← “Zn “+ZONA | | |
| LCD ← “Per “+PERSONAS | | |
| LCD ← “Par “+PARADAS | | |
| REFRESHTIMER ← millis() | | |
| . | | |
| LCD ← “” | | |
| LCD ← ACTIVESTR | | |
| LCD ← RTC | | |
| DISPTAR ← | | |
| TARIFA[PERSONAS-1]*10 | | millis()-refreshTimer>=500: han pasado más de 500 milisegundos desde el último refresco de displays & false: no modo OHPORDIOS & false: sistema no activo |
| LCD ← “Zn “+ZONA | | |
| LCD ← “Per “+PERSONAS | | |
| LCD ← “Par “+PARADAS | | |
| REFRESHTIMER ← millis() | | |
| . | | |
| LCD ← “” | | |
| LCD ← “OH POR DIOS!” | | |
| DISPTAR ← 666 | | |
| REFRESHTIMER ← millis() + | | millis()-refreshTimer>=500: han pasado más de 500 milisegundos desde el último refresco de displays & true: modo OHPORDIOS |
| 300 | | |
| . | | |
| • - | | millis()-refreshTimer<500: no han pasado más de 500 milisegundos desde el último refresco de displays |

Resultados

Material

- 1 tarjeta Arduino Mega
- 1 módulo RTC para Arduino
- 1 pantalla LCD de 16x2
- 1 display de 7 segmentos de 4 dígitos
- 1 potenciómetro de 5Ω
- 10 resistores de $10\,000\Omega$

- 4 resistor de $330\ \Omega$
- 1 resistor de $220\ \Omega$
- 10 push buttons
- 2 sensores magnéticos
- 1 protoboard pequeño
- 1 tabla de madera de $40 \times 40\text{cm}$ de 3mm de espesor
- 1 montura de succión para parabrisas
- 1 conector para encendedor de auto
- 1 plug para Arduino
- 4 imanes de neodimio de 1cm de diámetro
- 30 jumpers hembra-macho
- 20 jumpers macho-macho
- 10 metros de alambre calibre 22
- Cinta de aislar
- Cinta doble cara
- Soldadura de plomo/estaño
- 2 autos

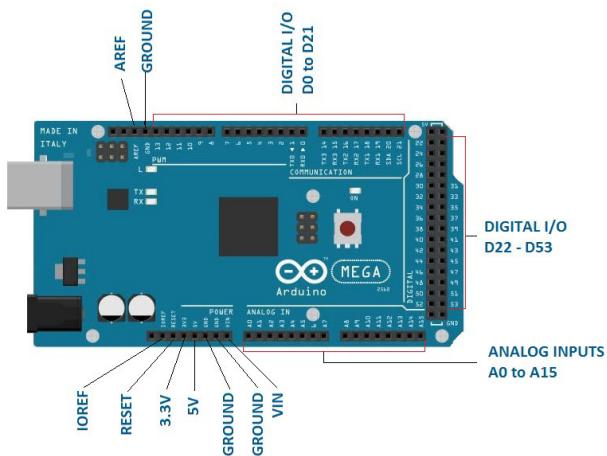


Figure 1: Arguino Mega

Desarrollo del circuito

En la siguiente figura se puede ver el circuito final, para desarrollarlo fue necesario planificar adecuadamente pues de otra manera los puertos del Arduino Mega no hubieran sido suficientes.



Figure 2: Vista frontal del taxímetro

Para la implementación del circuito se utilizó un Arduino Mega (código se encuentra abajo), al que se conectaron 10 push buttons, un sensor magnético y un reloj. Los push buttons fueron conectados a un puerto diferente cada uno, pero se conectaron los 10 a una misma tierra y un mismo VCC. Además, los sensores magnéticos están conectados a la tolva posterior del lado del copiloto de los coches y los imanes de neodimio se encuentran pegados a la parte interior del rin.

Para instalar el taxímetro en los autos de prueba se utilizó la montura de succión para parabrisas y se alimentó el circuito por medio del puerto para encendedor del auto. Éste provee 12V, y aunque el Arduino podría ser alimentado directamente por este voltaje, se usó un adaptador para bajar el voltaje a 6V.

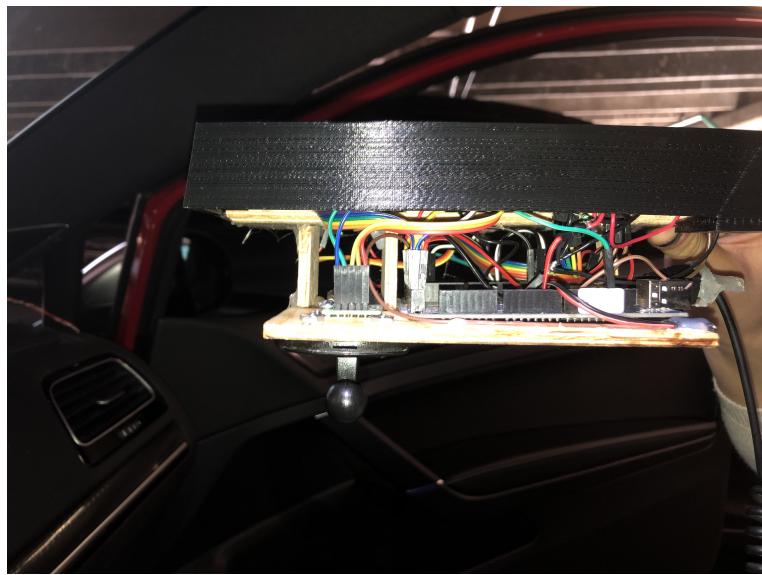


Figure 3: Cableado del circuito

Código desarrollado para la implementación del taxímetro

```
1  /*
2   * @AdeAndaK 173347
3   * @SLGonzAb 170404
4   *
5   * Código desarrollado para la implementación del cálculo de tarifa de un taxi con
6   * base en
7   * las variables de distancia, tiempo, horario, día de la semana, zona, personas,
8   * paradas y
9   * tipo de vehículo.
10  */
11
12  //Librerías utilizadas
13  #include <SevSeg.h>
14  #include <LiquidCrystal.h>
15  #include <ThreeWire.h>
16  #include <RtcDS1302.h>
17
18  //Declaración de objetos de librerías
19  SevSeg sevseg;
20  LiquidCrystal lcd(36, 37, 38, 39, 40, 41);
21  ThreeWire myWire(34,35,42); // DAT, CLK, RST
22  RtcDS1302<ThreeWire> Rtc(myWire);
23
24  //Pines de botones
25  int startStop = 44;
26  int zMas = 45;
27  int zMen = 46;
28  int OHPOR = 47;
29  int coche = 48;
30  int persMas = 49;
31  int paraMas = 50;
```

```

int diaMas = 53;
31 int horaMas = 52;
int horaMen = 51;
33
int sMag = 43; //Pin sensor magnetico
35
//Constantes
37 #define diaSec 86400; //segundos en un dia
#define horaSec 1200; //segundos en 20 minutos
39 #define banderaAuto 18; //tarifa base auto
#define banderaCami 25; //tarifa base camioneta
41 #define kmAuto 3.57; //tarifa por km en auto
#define kmCami 6.28; //tarifa por km en camioneta
43 #define minAuto 1.8; //tarifa por minuto en auto
#define minCami 3.15; //tarifa por minuto en camioneta
45 #define consAuto 7; //consumo promedio de combustible de auto
#define consCami 8.9; //consumo promedio de combustible de camioneta
47 #define diamAuto 0.00065; //diametro de llantas de auto en km
#define diamCami 0.00065; //diametro de llantas de camioneta en km
49 #define multHPico 1.5; //multiplicador de tarifa en hora pico
#define multFinDe 0.8; //multiplicador de tarifa en fin de semana
51 #define precioPara 4; //costo por parada

53 //Declaracion de variables de estado
bool active;
55 bool isCoche;
int zona;
57 bool modoDios;
int personas;
59 int paradas;

61 unsigned long startTime;
unsigned long vueltas;
63
//Declaracion de variables auxiliares
65 unsigned long refreshTimer;
RtcDateTime dt;
67 String activeStr;
String aux;
69 unsigned long bufferStart = 0;
unsigned long seconds2000;
71 double tarifa[4];
double distancia;
73 int dispTar;
double tarifaPar;
75 double bandera;
bool sMagAnt;
77 bool vueltaAct; //bandera de registro de vuelta en el ciclo actual
double kmVuelta;
79 double precioKm;
double precioMin;

```

```

81 unsigned long parCounter;
82 double consumoP;
83 unsigned int paraAnt;
84 unsigned int contadorDios;
85
86 void setup() {
87     //Configuracion de 7segmentos
88     byte numDigits = 4;
89     byte digitPins[] = {30, 32, 31, 33};
90     byte segmentPins[] = {29, 27, 25, 23, 22, 28, 26, 24};
91     bool resistorsOnSegments = true;
92     bool updateWithDelaysIn = true;
93     byte hardwareConfig = COMMON_CATHODE;
94     sevseg.begin(hardwareConfig, numDigits, digitPins, segmentPins,
95         resistorsOnSegments);
96     sevseg.setBrightness(70);
97
98     //Configuracion de LCD
99     lcd.begin(16, 2);
100
101     //Configuracion RTC
102     //Serial.begin(9600);
103     Rtc.Begin();
104     RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
105
106     //Rtc.SetDateTime(compiled);
107
108     if (!Rtc.IsDateTimeValid()){
109         // Common Causes:
110         // 1) first time you ran and the device wasn't running yet
111         // 2) the battery on the device is low or even missing
112
113         //Serial.println("RTC lost confidence in the DateTime!");
114         Rtc.SetDateTime(compiled);
115     }
116     if (Rtc.GetIsWriteProtected()){
117         //Serial.println("RTC was write protected, enabling writing now");
118         Rtc.SetIsWriteProtected(false);
119     }
120     if (!Rtc.GetIsRunning()){
121         //Serial.println("RTC was not actively running, starting now");
122         Rtc.SetIsRunning(true);
123     }
124     RtcDateTime now = Rtc.GetDateTime();
125     if (now < compiled){
126         //Serial.println("RTC is older than compile time! (Updating DateTime)");
127         Rtc.SetDateTime(compiled);
128     }
129     else if (now > compiled){
130         //Serial.println("RTC is newer than compile time. (this is expected)");
131     }

```



```

181     parCounter=millis();
182     vueltas=0;
183     distancia=0;
184     tarifa[1]=0;
185     tarifa[2]=0;
186     tarifa[3]=0;
187     if(isCoche) {
188         bandera=banderaAuto;
189         kmVuelta=diamAuto;
190         precioKm=kmAuto;
191         precioMin=minAuto;
192         consumoP=consAuto;
193     }else{
194         bandera=banderaCami;
195         kmVuelta=diamCami;
196         precioKm=kmCami;
197         precioMin=minCami;
198         consumoP=consCami;
199     }
200     tarifa[0]=bandera;
201     kmVuelta*=PI;
202     //Serial.println("Distancia por vuelta "+(String)(kmVuelta*1000));
203 }else{
204     activeStr="Inactivo";
205     personas=1;
206 }
207     bufferStart=millis();
208 }
209 if(digitalRead(horaMas)==1 && millis()-bufferStart >= 400){
210     lcd.setCursor(0,0);
211     if(active){
212         lcd.print("Bloqueado      ");
213     }else{
214         lcd.print("20min+  ");
215         RtcDateTime dtH=Rtc.GetDateTime();
216         seconds2000=dtH.TotalSeconds() + horaSec;
217         dtH=RtcDateTime(seconds2000);
218         Rtc.SetDateTime(dtH);
219     }
220     bufferStart = millis();
221     refreshTimer=millis()-100;
222 }
223 if(digitalRead(horaMen)==1 && millis()-bufferStart >= 400){
224     lcd.setCursor(0,0);
225     if(active){
226         lcd.print("Bloqueado      ");
227     }else{
228         lcd.print("20min-  ");
229         RtcDateTime dtH=Rtc.GetDateTime();
230         seconds2000=dtH.TotalSeconds() - horaSec;
231         dtH=RtcDateTime(seconds2000);

```

```

        Rtc.SetDateTime(dtH);
233    }
    bufferStart = millis();
235    refreshTimer=millis()-100;
}
237 if(digitalRead(diaMas)==1 && millis()-bufferStart >= 400) {
    lcd.setCursor(0,0);
239    if(active){
        lcd.print("Bloqueado      ");
241    }else{
        lcd.print("Dia+      ");
243    RtcDateTime dtD=Rtc.GetDateTime();
        seconds2000=dtD.TotalSeconds() + diaSec;
245    dtD=RtcDateTime(seconds2000);
        Rtc.SetDateTime(dtD);
247    }
    bufferStart = millis();
249    refreshTimer=millis()-100;
}
251 if(digitalRead(zMas)==1 && millis()-bufferStart >= 400) {
    lcd.setCursor(0,0);
253    lcd.print("Zona+      ");
    zona+=1;
255    if(zona==5) {
        zona=1;
257    }
    bufferStart = millis();
259    refreshTimer=millis()-100;
}
261 if(digitalRead(zMen)==1 && millis()-bufferStart >= 400) {
    lcd.setCursor(0,0);
263    lcd.print("Zona-      ");
    zona-=1;
265    if(zona==0) {
        zona=4;
267    }
    bufferStart = millis();
269    refreshTimer=millis()-100;
}
271 if(digitalRead(coche)==1 && millis()-bufferStart >= 400) {
    lcd.setCursor(0,0);
273    if(active){
        lcd.print("Bloqueado      ");
275    }else{
        isCoche=!isCoche;
277        if(isCoche){
            lcd.print("SLGA      ");
279        }else{
            lcd.print("ADAKU      ");
281        }
    }
}

```

```

283     bufferStart = millis();
284     refreshTimer=millis();
285 }
286 if(digitalRead(persMas)==1 && millis()-bufferStart >= 400) {
287     lcd.setCursor(0,0);
288     lcd.print("Pers+          ");
289     if(active && personas!=4) {
290         tarifa[personas]+=bandera;
291     }
292     if(personas==4 && !active) {
293         personas=1;
294     }else if(personas!=4) {
295         personas+=1;
296     }
297     bufferStart = millis();
298     refreshTimer=millis();
299 }
300 if(digitalRead(paraMas)==1 && millis()-bufferStart >= 400) {
301     lcd.setCursor(0,0);
302     if(active){
303         paradas+=1;
304         lcd.print("Para+          ");
305     }else{
306         lcd.print("Bloqueado");
307     }
308     bufferStart = millis();
309     refreshTimer=millis();
310 }
311 //Calculo de tarifa parcial
312 if(active){
313     vueltaAct=false;
314     tarifaPar=0;
315     if(digitalRead(sMag)==1 && !sMagAnt) {
316         //se registro una vuelta y en el ciclo anterior el sensor estaba en bajo
317         vueltas+=1;
318         vueltaAct=true;
319         sMagAnt=true;
320         //Serial.println("Vuelta");
321     }else if(digitalRead(sMag)==0 && sMagAnt) {
322         //el sensor esta en bajo pero estuvo en alto el ciclo anterior
323         sMagAnt=false;
324     }
325     if(vueltaAct){
326         tarifaPar+=precioKm*kmVuelta;
327     }
328     tarifaPar+=(millis()-parCounter)*precioMin/60000;
329     parCounter=millis();
330 }
331     dt=Rtc.GetDateTime();
332     tarifaPar*=multPico(dt);

```

```

    tarifaPar*=multDia(dt);
335  tarifaPar*=multZona();
    distancia=vueltas*kmVuelta;
337  if(distancia/((millis()-startTime)/1000<=10) && millis()-startTime/60000<=10) {
        //si la velocidad promedio es menor a 10km/h y el tiempo de viaje es mayor a 10
        minutos
        if(distancia/((millis()-startTime)/1000<=5)) {
339    tarifaPar*=2;
        }else{
341    tarifaPar*=100/(distancia/((millis()-startTime)/1000)*consumoP));
        }
343  }
    if(paradas>paraAnt) {
345    tarifaPar+=(paradas-paraAnt)*precioPara;
        paraAnt=paradas;
347  }
    for (int i = 0; i < personas; i++) {
349    tarifa[i]+=tarifaPar*((7-personas)/6.0); //suma a la tarifa por persona
        }
351 }

353 //Mostrar informacion en displays LCD y 7seg
if(millis()-refreshTimer >= 500) {
355    lcd.clear();
    lcd.setCursor(0,0);
357    if(!modoDios){
        lcd.print(activeStr);
359    if(active){
            distancia=vueltas*kmVuelta;
361    lcd.print((String)distancia+"km ");
            lcd.setCursor(10,0);
363    lcd.print((String)((millis()-startTime)/60000)+"min");
            dispTar=tarifa[0]*10;
365    }else{
            lcd.setCursor(11,0);
367    dt=Rtc.GetDateTime();
            aux="";
369    if(dt.Minute()<10){
                aux="0";
            }
            lcd.print((String)dt.Hour()+":"+aux+dt.Minute());
373    lcd.setCursor(15,1);
            lcd.print(letraDia(dt));
375    dispTar=tarifa[personas-1]*10;
            }
377    sevseg.setNumber(dispTar,1);
            lcd.setCursor(0,1);
379    lcd.print("Zn "+letraZona(zona)+"/"+Per+(String)personas+"/"+Par+(String)
            paradas);
            refreshTimer=millis();
381    }else{

```

```

383     lcd.print("OH POR DIOS!, NO");
384     lcd.setCursor(0,1);
385     lcd.print("HAY VUELTA ATRAS");
386     sevseg.setNumber(666,contadorDios%4);
387     contadorDios++;
388     refreshTimer=millis()+300;
389 }
390     sevseg.refreshDisplay();
391 }

393 String letraDia(RtcDateTime dt){
394     int dia=dt.DayOfWeek();
395
396     switch (dia){
397         case 0:
398             return "D";
399             break;
400         case 1:
401             return "L";
402             break;
403         case 2:
404             return "M";
405             break;
406         case 3:
407             return "W";
408             break;
409         case 4:
410             return "J";
411             break;
412         case 5:
413             return "V";
414             break;
415         case 6:
416             return "S";
417             break;
418     }
419     return "I";
420 }
421
422 String letraZona(int zn){
423     switch (zn){
424         case 1:
425             return "G";
426             break;
427         case 2:
428             return "A";
429             break;
430         case 3:
431             return "C";
432             break;

```

```

433     case 4:
434         return "P";
435     break;
436 }
437 return "X";
438 }

439 double multZona(){
440     switch(zona){
441         case 1:
442             return 1;
443             break;
444         case 2:
445             return 1.5;
446             break;
447         case 3:
448             return 1.3;
449             break;
450         case 4:
451             return 1.2;
452             break;
453     }
454     return 1;
455 }
456 }

457 double multPico(RtcDateTime dtP){
458     dtP=Rtc.GetDateTime();
459     if((dt.Hour()>6 && dt.Hour()<10) || (dt.Hour()>17 && dt.Hour()<20)){
460         return multHPico;
461     }
462     return 1;
463 }
464 }

465 double multDia(RtcDateTime dtF){
466     dtF=Rtc.GetDateTime();
467     if(dt.DayOfWeek()==6 || dt.DayOfWeek()==0){
468         return multFinDe;
469     }
470     return 1;
471 }

```

tarifador_V1/tarifador_V1.ino

Análisis de resultados

En la mayor parte de las pruebas realizadas se obtuvieron los resultados esperados, pues la tarifa y los parámetros mostrados corresponden con los cálculos realizados y con los valores reales de tiempo y distancia. Esta correspondencia se cumple para las primeras tres etapas de pruebas y la primera fase de la cuarta etapa. En la última fase de pruebas se realizaron

pruebas a velocidades por encima de 70 km/h, en las que se pudo observar un valor de distancia registrado entre 10% y 20% menor al valor real.

Este error se puede ocasionar por diversos factores y a continuación se discuten junto con las posibles soluciones. En primer lugar, se puede deber a que el sensor magnético no tiene suficiente tiempo para detectar el paso del imán. Esto podría solucionarse al colocar el arreglo de sensor-imán más cerca del centro de la rueda, lo que reduciría la velocidad del imán con respecto al sensor sin modificar el número de vueltas registradas. En segundo lugar puede ser consecuencia de interferencia entre los alambres del sensor, pues recorren una distancia relativamente larga y a altas velocidades el valor del sensor cambia rápidamente, lo que se podría corregir separando físicamente los alambres. Por último, puede deberse a un error inherente a la utilización de un sensor magnético; este problema podría ser corregido haciendo uso del puerto OBDII del auto para obtener datos de velocidad y revoluciones del motor. Esta solución fue considerada, pero se descartó debido al aumento considerable de costos por el adaptador necesario para la comunicación OBDII-Arduino.

Conclusión

El proyecto realizado presentó una serie de retos inesperados de todas las áreas involucradas, pues antes de empezar con el desarrollo tangible se tuvo que planear a detalle el material necesario y el itinerario de trabajo. Esto fue de gran importancia pues los puertos digitales del Arduino Mega fueron aprovechados en su totalidad, lo que puede verse tanto como una limitante de hardware o como una optimización de los recursos disponibles.

Más adelante se tuvieron que resolver problemas propios de electrónica, pues se planearon las redes de entradas y salidas, así como el cableado de los displays utilizados y el cálculo de los resistores asociados. Posterior a la etapa de cableado y diseño físico de la caja impresa en 3D (modelada en NX), se pasó a la implementación del código para resolver el problema, en la que se consultaron ampliamente las referencias de los componentes utilizados y se aprendió en el proceso. Por último, al instalar el sistema en los autos de prueba se tuvieron que considerar no sólo las partes móviles, pero también las fuentes de calor como el sistema de frenado; y también se planteó el problema de trabajar de manera segura debajo de un vehículo.

En general se pudo aprender de este proyecto más allá de los temas de la materia de Circuitos Lógicos con experiencias y problemas tangibles. Consideramos que fue de gran utilidad para poner a prueba la capacidad de resolver problemas nuevos con los conocimientos adquiridos hasta el momento y unos cuantos más que fue necesario investigar.

Referencias

1. Knap.at. (2019). [online] Available at: <http://www.knap.at/datenblaetter/led/> [Accessed 11 Nov. 2019]
2. SURTR TECHNOLOGY. (2019). How to simply use DS1302 RTC module with Arduino board and LCD screen. [online] Available at: <https://surtrtech.com/2018/01/>

27/how-to-simply-use-ds1302-rtc-module-with-arduino-board-and-lcd-screen/ [Accessed 11 Nov. 2019].

3. Pattabiraman, K. (2019). How to Set up 7-Segment Displays on the Arduino - Circuit Basics. [online] Circuit Basics. Available at: <http://www.circuitbasics.com/arduino-7-segment> [Accessed 11 Nov. 2019].
4. Forward.com.au. (2019). How to code Timers and Delays in Arduino. [online] Available at: <https://www.forward.com.au/pfod/ArduinoProgramming/TimingDelaysInArduino.html> [Accessed 11 Nov. 2019].
5. Circuitstoday.com. (2019). [online] Available at: <http://www.circuitstoday.com/wp-content/uploads/2014/06/interfacing-LCD-to-arduino.png> [Accessed 11 Nov. 2019].
6. Arduino.cc. (2019). Arduino - HelloWorld. [online] Available at: <https://www.arduino.cc/en/Tutorial>HelloWorld> [Accessed 11 Nov. 2019].
7. GitHub. (2019). Makuna/Rtc. [online] Available at: <https://github.com/Makuna/Rtc/wiki/RtcDateTime-object> [Accessed 11 Nov. 2019].
8. Arduino.cc. (2019). Arduino Reference. [online] Available at: <https://www.arduino.cc/reference/en/#page-title> [Accessed 11 Nov. 2019].

Adicionalmente, se puede consultar el código completo en <https://github.com/adeandak/taximetro>