14B

Arduino UNO Based Taximeter

14B.1 Introduction

(1) A Taximeter is an electronic instrument (Fig-14B.1) which is fitted with a Taxicab to record: Distance travelled, Waiting time, and Fare.

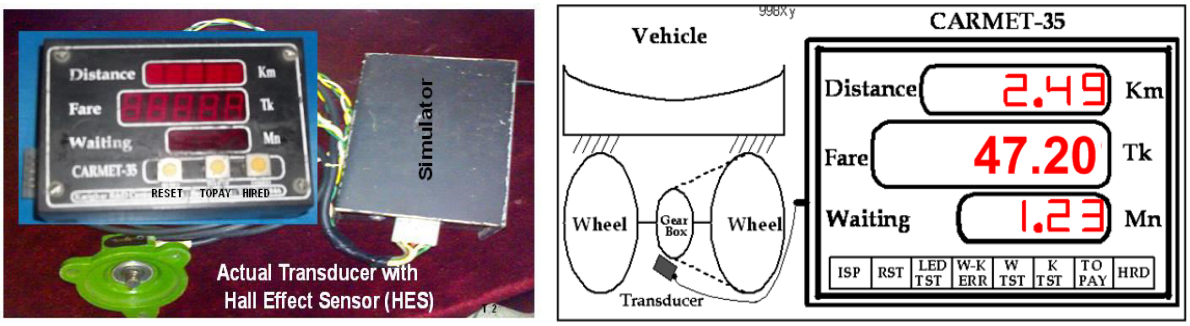

Figure-14B.1: *Pictorial/Conceptual view of a 3-widow Taximeter*

(2) **Working Principle:** The green color gadget of Fig-14B.1 is the transducer which is fitted with the gearbox of the Taxicab, and it generates about 16 pulses when the Taxicab moves by 10 m distance. These pulses are called WTP (wheel turning pulses), and they are accumulated by TC0 Module of the MCU to update DDM Meter (Digital Distance Meter) by 10 m.

The flat fare for the first 2 km is Tk 40.00 and then the fare is Tk 12.00 for each km covered. The DFM Meter (Digital Fare Meter) accumulates/updates fare for each 200 m travelled which is Tk 2.40. When the Taxicab just crosses initial 2 km distance, the DFM shows a reading of Tk 14.40 (12.00 + 2.40) which means that the Taxi driver is paid an advance payment for 200 m; as a result, there is no chance for the driver to lose fare even the passenger gets down at any place.

There is a fare for waiting time which is Tk 2.00 for each minute waiting. As the practical journey/trip is a combination of distance travelled and waiting time, the fare is always computed based on the combined effect of distance and waiting time; where distance is the **reference.** The DWM Meter (Digital Wait Meter) records the total waiting time happened in the trip (0.00 to 99.99 = 0 Min 0 Sec to 99 Min 99 Sec).

There are three buttons on the face plate of the Taximeter of Fig-14B.1 and these are: RESET, TOPAY, and HIRED. The RESET button brings 0.00 of DDM, 0.00 of DFM, and 0.00 of DWM. When the Taxicab is hired, the driver pushed the HIRED button; as a result, the DFM shows 40.00 — the flat fare.

(3) There could be three types of journey/trip for the taxicab and these are:

- **(a)** **Distance-drive (**D-drive**):** There is no waiting in the trip. The whole fare is due to distance travelled and the flat fare.
- **(b)** **Time-**drive (T-drive): There is no distance travelled. The whole fare is due to waiting time and the flat fare.
- **(c)** **Combined-drive** (C-drive): This is the practical journey. The whole fare is due to the combined effect of distance travelled, waiting time, and flat fare.

14B.2 Program for D-drive Journey

We will simulate WTP pulses using 555-based 1 Hz oscillator. The DDM, DFM, and DWM Meters will be built using 5xcc7seg display devices in multiplexed mode. The hardware setup would be around a repackaged Arduino UNO Board (Fig-14B.2). The schematic diagram for DFM is shown in Fig-14B.3. In this Section, we will develop sketch for implementation of D-drive journey depicted in Fig-14B.4–Fig-14B.6.

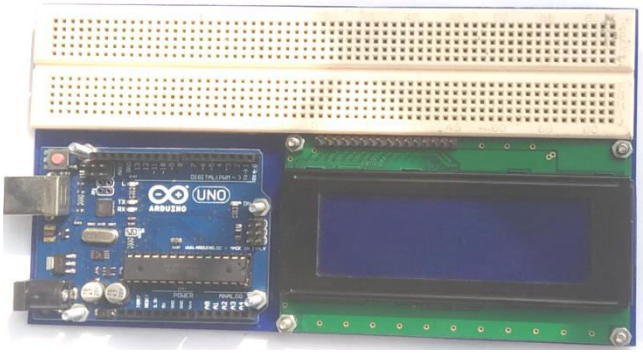

Figure-14B.2: *Repackaged version of Arduino UNO Board*

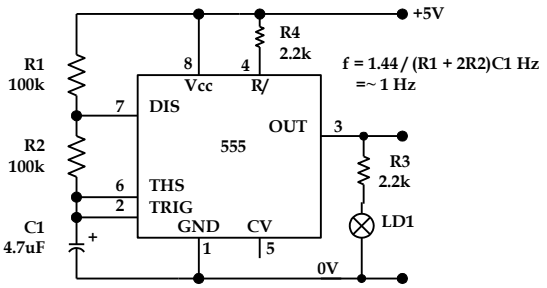

Figure-14B.3: *555-based 1 Hz oscillator*

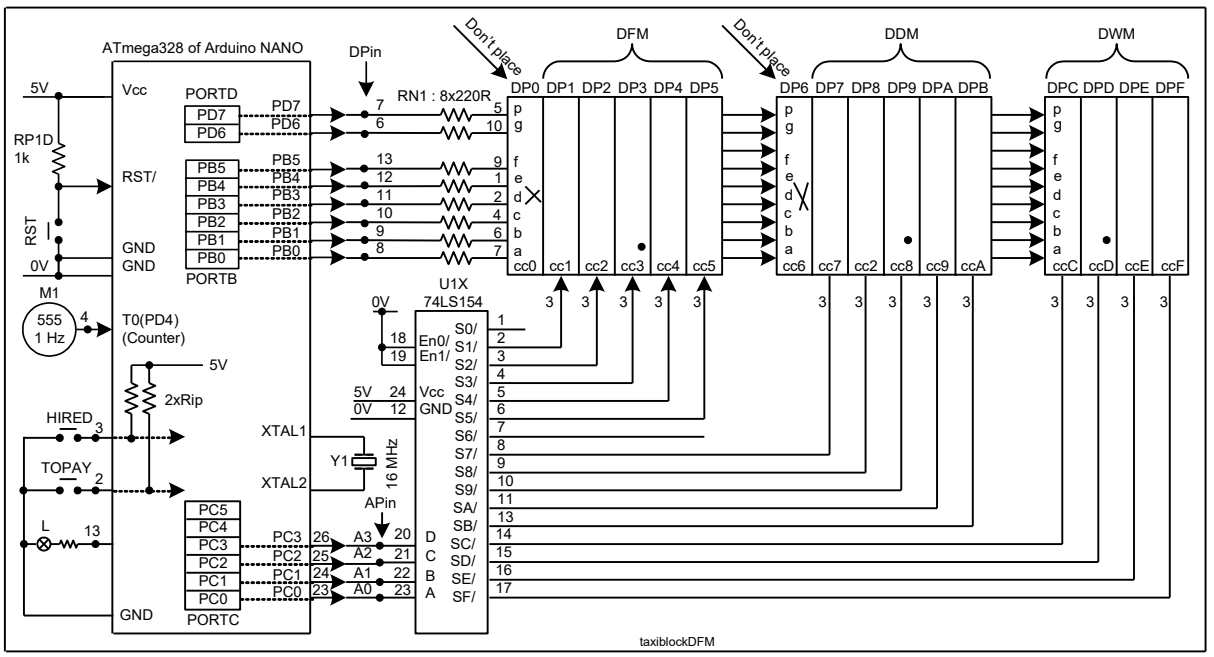

Figure-14B.4: *Schematic Diagram for the Taximeter*

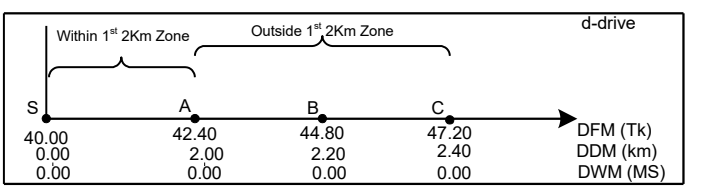

Figure-14B.5: *Line diagram of a typical D-drive journey*

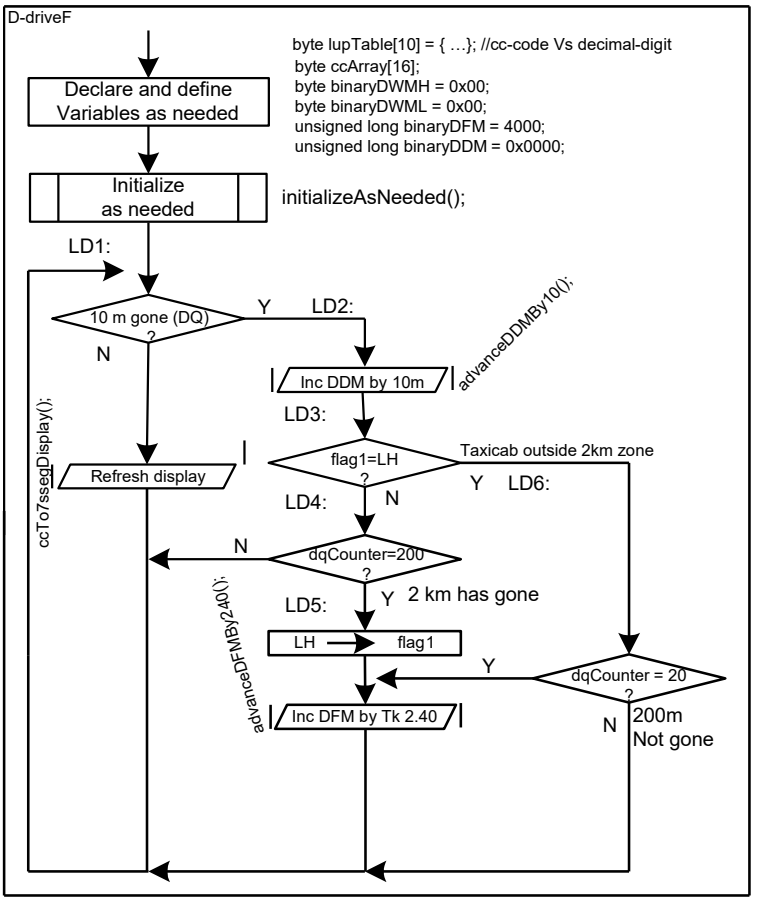

Figure-14B.6: *Flow Chart describing the algorithm for the D-drive journey of Fig14B.5*

(1) Place the following display devices on the breadboard: DP1 – DP5 for DFM, DO7 – DPB for DDM, and DPC – DPF for DWM. Connect them together and with UNO as per Fig-14B.4 to form multiplexed display unit.

(2) Place U1X (4-to-16 decoder) at the rightmost side of the breadboard and connect it with UNO and display unit as per Fig-14B.4.

14B.3 Display Testing Program for D-drive Journey

(1) Create and upload a sketch to show 987.65 at DP1 – DP5 positions of DFM. The point (.) will appear at DP3 position with 7 (7.). The following hints could be helpful.

(2) The display devices are cc-type; therefore, we need to declare a lookup table (LUT) holding cc-codes for the digits 0 – 9.

```
byte lupTable[17] = { 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F};
```

(3) Let us keep in mind that the decimal point is always before 2-digit from the right of the given number 987.65 (or any other number). Now, we have simply 98765 (a decimal number) which when stored in memory will appear as: 0x181CD (> 16-bit). Let us declare the following variables for the storage of the data of DFM, DDM, and DWM.

```
unsigned long binaryDFM;      //Fare may range: Tk 000.00–999.99 (0x00000000 – 0x0001869F)
unsigned long binaryDDM;      //distance may range: km 000.00–999.99(0x00000000 = 0x0001869F)
byte binaryDWMH;              //Minute of Digital Wait Meter may range: 00 – 99 (0x00 – 0x63)
byte binarytDWL;              //Second of DWM may range: 00 – 59 (0x00 – 0x3B)
```

(4) **Let** us use the following data structure of Fig-14B.7 to handle data exchange between UNO and display unit.

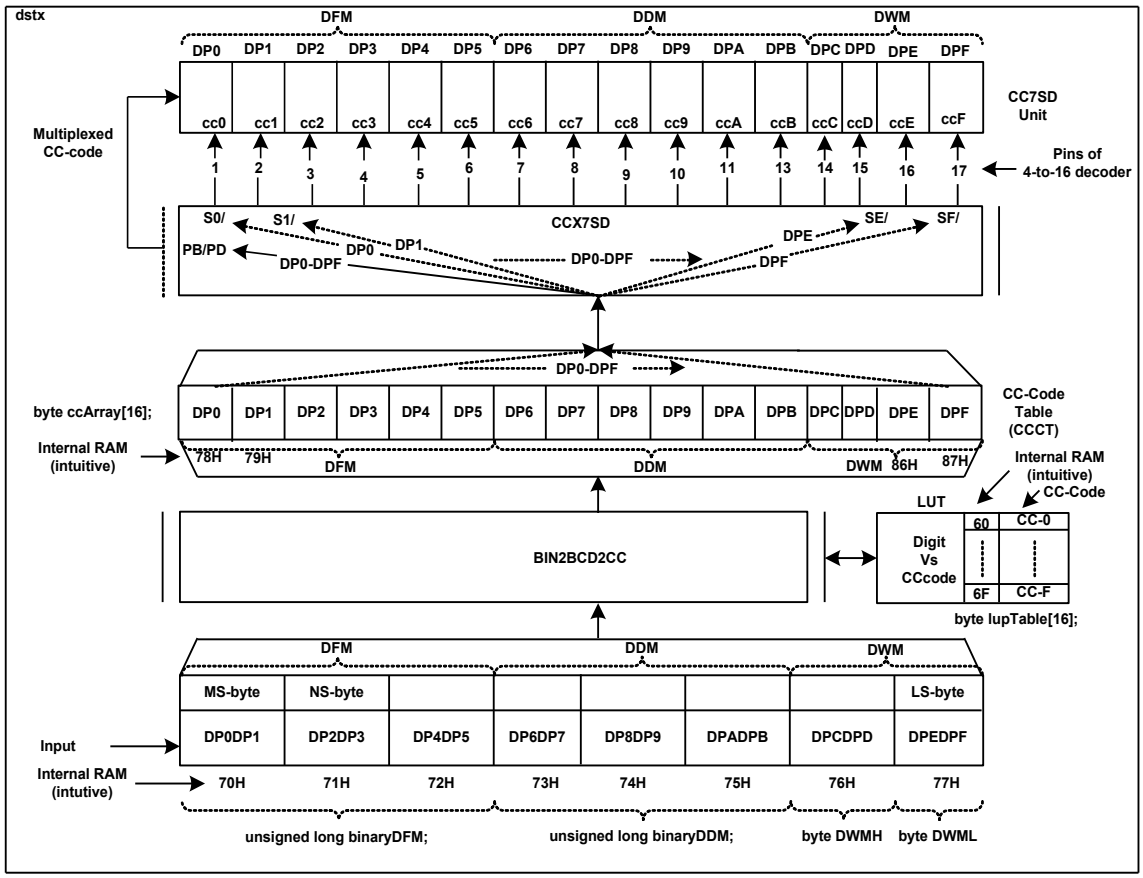

Figure-14B.7: *Data structure for the Taximeter*

(5) Store the given test fare 987.65 in the variable `bunaryDFM` of Fig-14B.7; get the indices (05, 06, 07, 08, 09) for the digits of the fare; use these indices to collect the corresponding cc-code from lookup table of Step-3a and save them into ccArray[ ].

```
unsigned long binaryDFM = 978.65;       //DFM will automatically contain: 0x0001869F
byte ccArray[6];                        //6-bye cc-codes for DFM (only 5 bytes are active)
byte x;
for(int i = 5; i>=0; i--)
{
     x = DFM % 10;                  //get the index for LS-digit (05)
     ccArray[i] = lupTable[x];      //collect cc-code for 5 and save in ccArray[5]
     DFM = DFM / 10;                //leave the quotient for next index
}
```

(6) In Step-5, we have computed 6 indices; now, we have 6 units of cc-codes in the ccArray[] for the positions: DP0 – DP5. Display at DP0 position is absent; so, nothing will appear there.

(7) Add point (.) with the content of DP3 position and then transfer the content of ccArray[] to DFM Meter of the display unit.

```
void ccTo7SegDisplay()
{
     ccArray[3] = ccArray[3] | 0x80;    //adding point (.)
     for (int j=0; j<5; j++)
     {
```

```
            byte x = ccArray[j];
            PORTB = x;
            digitalWrite(6, bitRead(x, 6));
            digitalWrite(7, bitRead(x, 7));
            PORTC = j;                          //enable DP0, DP1, …, DP5
            delay(5);
        }
}
```

(8) Based on hints given above, develop sketch, upload and check that DFM shows: 987.65.

(9) Bring necessary change in the sketch of Step-4 to show 40.00 in the DFM of the display. Note that the leading 0s are to be suppressed.

14B.4 Driving DDM and DFM for D-drive Journey

(1) Place a 555 IC on the breadboard of Fig-14B.2 and build the oscillator as per Fig-14B.3. Connect the output of the oscillator with DPin-4 of UNO from which the simulated WPT pulses will be entering into the TC0 Module of the ATmega328P MCU.

(2) Configure TC0 Module to count external pulses via DPin-4 and when the accumulated count will be equal to 16, the DDM will be advanced by 10 m. When the DDM will just cross 2.00 km, the flat fare 40.00 will be augmented by Tk 2.40. For subsequent coverage of each 200 m distance, the fare will be incremented by Tk 2.40. This feature of the D-drive has been depicted in Fig-14B.5 and 14B.6.

(3) Create and upload sketch to accomplish the following tasks:

(a) Disconnect 1 Hz oscillator for DPin-4. Press RST button of UNO; check that the following reading have appeared on the display unit.

DFM : 0.00
DDM : 0.00
DWM : 0.00

(b) Connect HIRED button with UNO as per Fig-14B.4. Press RST Button and then HIRED Button. The display will show:

DFM : 40.00
DDM : 0.00
DWM : 0.00

Codes:

```
4000      → binaryDFM
0         → binaryDDM
0         → binaryDWMH
0         → binaryDWML
binaryToIndexToCC();          //get indices for the digit; get cc-cdoes and save
suppressLZeros();             //suppressed leading zeros
ccTo7segDisplay();            //show all meter readings on diaply
```

(4) Connect 1 Hz oscillator at DPin-4 of UNO. Check that for each 10 m travel of the Taxicab, the DDM advances by 10 m. The DFM shows 42.40 when the DDM just shows: 2.00 km. Let us assume that there is no waiting in the journey.

```
void loop()
{
     while(10 m has not elapsed)
     {
          Refresh multiplexed display unit by calling ccTo7SegDisplay().
     }
     DDM = DDM + 0.01;
     if(DDM = 2.00 km)
     {
          DFM = DFM + 2.40
     }
}
```