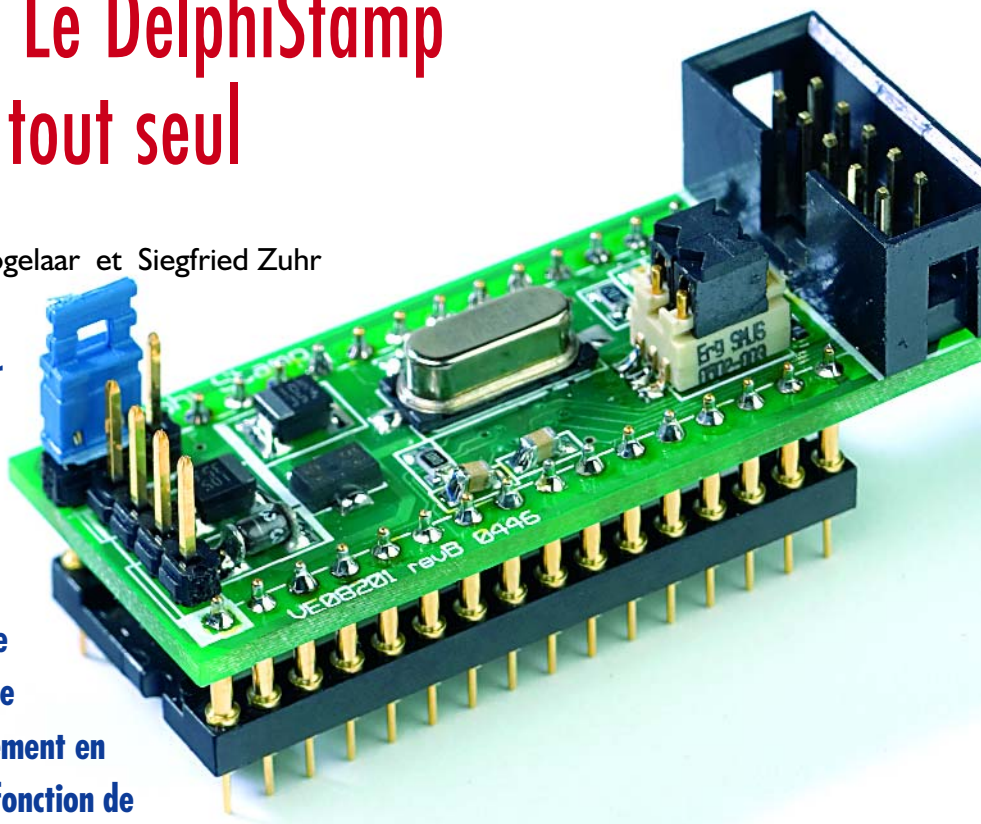
DELPHI POUR ÉLECTRONICIENS

Suite & Fin – Le DelphiStamp se débrouille tout seul

Detlef Overbeek, Anton Vogelaar et Siegfried Zuhr



Nous allons, dans ce dernier article du cours Delphi, broder sur le projet d'arroseur automatique décrit dans la 9ème partie. Nous augmentons le nombre de vannes et le dotons d'une pompe pour l'approvisionnement en eau. Nous lui ajoutons une fonction de timer. Le résultat est un système d'arrosage complet qui pourra vous servir dès l'été prochain.

Dans l'article du mois dernier nous avons démarré un projet « d'Arroseur arrosé ». Nous avions créé un concept de base, suivi d'une simulation sous Delphi, le programme résultant étant ensuite placé dans le DelphiStamp. Nous avions utilisé pour cela la carte d'évaluation correspondante et surtout l'affichage LCD, le bouton-poussoir, un second bouton (connecté aux contacts de la seconde entrée analogique) et les LED servant à l'identification des sorties pilotées.
Il est possible de sélecter certaines vannes, de définir la durée d'ouverture d'une vanne et de fixer le nombre de répétition du cycle.
Le but de cet article est de doter le projet d'un nombre de vannes plus important, de lui permettre de piloter une pompe et de pouvoir démarrer un cycle à un moment quelconque (la nuit par exemple).
Pour cela nous voulons pouvoir interroger le paramétrage et le modifier via le PC par le biais d'un programme de suivi et de configuration.
(Le programme pour le DelphiStamp est développé sous Delphi, compilé à l'aide du compilateur croisé et chargé dans le DelphiStamp. Ce dernier fonctionne de façon totalement autonome.
Parallèlement, le programme de suivi/configuration constitue une application distincte. Ce programme tournant sur le PC, il devient possible, par le biais du câble RS-232, de lire les paramètres du programme dans le DelphiStamp et de les modifier, comme s'il s'agissait d'une sorte de télécommande).

Cette extension peut se faire selon 2 trajets :
Soit nous mettons en oeuvre les E/S présentes sur le DelphiStamp, soit nous utilisons le port I²C pour communiquer avec l'extension.
Nous avons ici opté pour l'interface I²C, les boutons-poussoirs et sorties requis sont connectés au DelphiStamp, par le biais d'un circuit d'E/S. Nous avons fait appel pour cela à un PCF8574, circuit comportant 8 lignes d'E/S qu'il est possible de lire et sur lesquelles on peut écrire. Il est possible d'adresser un maximum de 16 de ces circuits intégrés sur le bus, ce qui signifie qu'il est possible, aisément, d'étendre le circuit jusqu'à un maximum de 16 x 8 = 128 lignes d'E/S.
Si l'on veut, ultérieurement, réaliser une application auto-

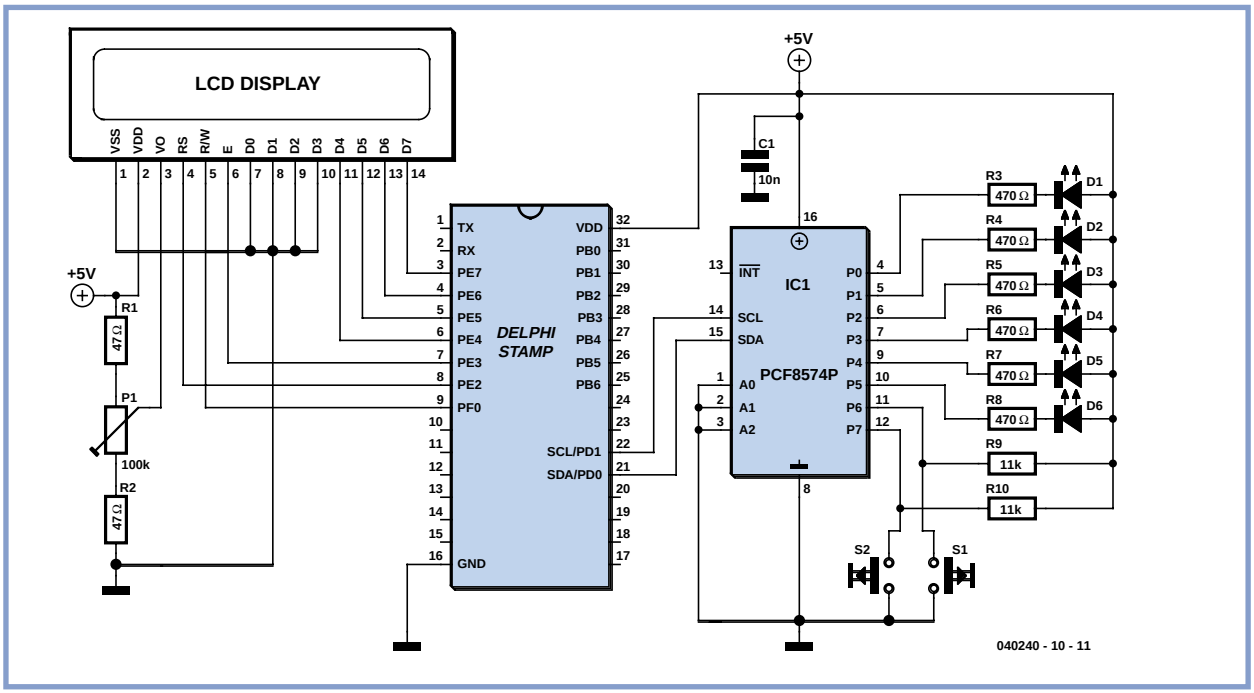


Figure 1. Le schéma de l'électronique objet de cette dernière partie du cours.

nome (ne comportant plus la carte d'évaluation), tout ce dont nous avons besoin, outre du DelphiStamp, sont les lignes vers l'affichage LCD, le circuit d'E/S et les composants de puissance (**figure 1**).

Le logiciel

Nous utilisons, comme base de développement du logiciel, le code source de l'article du mois dernier. Il faut ajouter à la simulation la puce d'E/S soit 2 boutons-poussoirs et 6 sorties. Ils sont placés dans l'Interface Graphique Utilisateur (le panneau de commande) en-dessous de ce qui existe déjà (**figure 3**). Dans le code sousjacent nous procédons à la modification de certains points.

Le menu et sa structure

Nous plaçons 2 boutons sur le forme du programme. Ce seront des SpeedButtons (vu qu'ils ont une caractéristique Down; après un clic nous faisons passer cette propriété Down à True).
Nous ajoutons, à l'unité Udrivers, 2 procédures et une fonction, une TWI_init (Two Wire Interface = I²C), un PCF8574_Wr (Write) et PCF8574_Rd (Read). Ceux-ci assurent, dans la simulation, le traitement que permettent les pilotes intégrés dans le DelphiStamp. Dans la simulation la procédure init ne fait rien, elle reste par conséquent vide.
Le PCF8574_Wr sert à faire changer, sur l'écran, la couleur de la LED requise. Le PCF8574_Rd suit les actions sur les boutons-poussoirs. Ceci est fait par une interrogation de la propriété Down pour voir si elle se trouve à True. Si c'est le cas, il en est tenu compte dans la valeur du nombre que la fonction renvoie.
Nous en avons terminé avec l'extension de la simulation. Le coeur du programme, l'unité UControl, requiert plus d'attention. Nous voulons faire passer le nombre de vannes de 3 à 5, la 6ème sortie nous servant à piloter une pompe de manière à disposer de notre propre pression. Il nous faut aussi ajouter une ligne de menu pour pouvoir

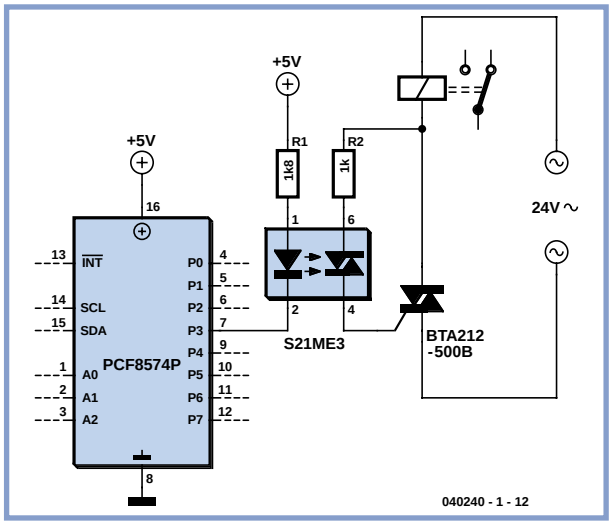


Figure 2. Exemple de pilotage d'une vanne à l'aide d'une tension 24 V alternatifs.

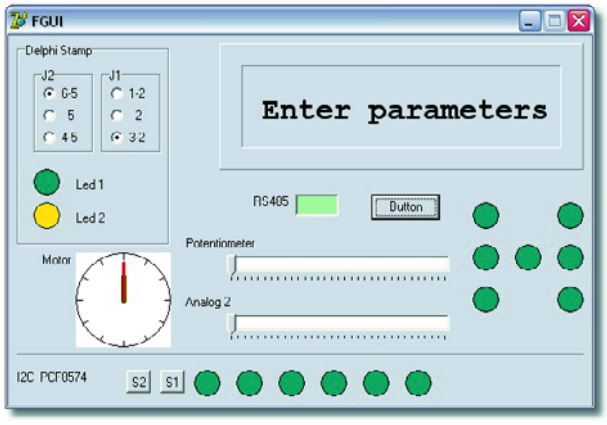


Figure 3. Dans le bas à gauche : les 2 boutons et les 6 LED.

définir l'instant de mise en route.
Pour cela, nous faisons passer à 10 positions la matrice qui tourne parallèlement au menu.
Dans le code du menu il faut tenir compte de la nouvelle longueur de la matrice. Nous intercalons la ligne de menu « Starting time » entre « Number of cycles » et la première vanne (**figure 4**). Il nous est possible ainsi de numéroter les vannes dans l'ordre en cas d'ajout de nouvelles vannes, sans qu'il ne soit nécessaire de modifier la structure du menu.
L'option de menu « Starting » est dotée d'un sous-menu. Nous avons le choix entre « Start direct » et « Start delayed ». Un « Exit » nous permet de revenir au menu principal.
Nous retrouvons un menu similaire derrière l'option de menu « Starting time », avec les options « Set start hour » « Set start minute » et « Exit ».
Nous ajoutons, pour le traitement des textes sur l'affichage LCD, les procédures « ShowStart » et « ShowTimer ».
Lors du traitement du menu dans le sens vertical (par le bouton 1) nous adaptons Menulevel 1. Il est doté d'un sous-menu à 3 options
De plus, le traitement des vannes est décalé d'une position, passant de >4 à >5 dans la comparaison {If MenuLevel > 5 then}. On trouve alors en position 5 le traitement de l'augmentation de la valeur heure/minute. En cas de dépassement de 23 heures, la variable repasse à 0 heure et pour les minutes on repasse à 0 lors du dépassement de 59.
L'heure est stockée dans 2 variables, S_hour et S_min.

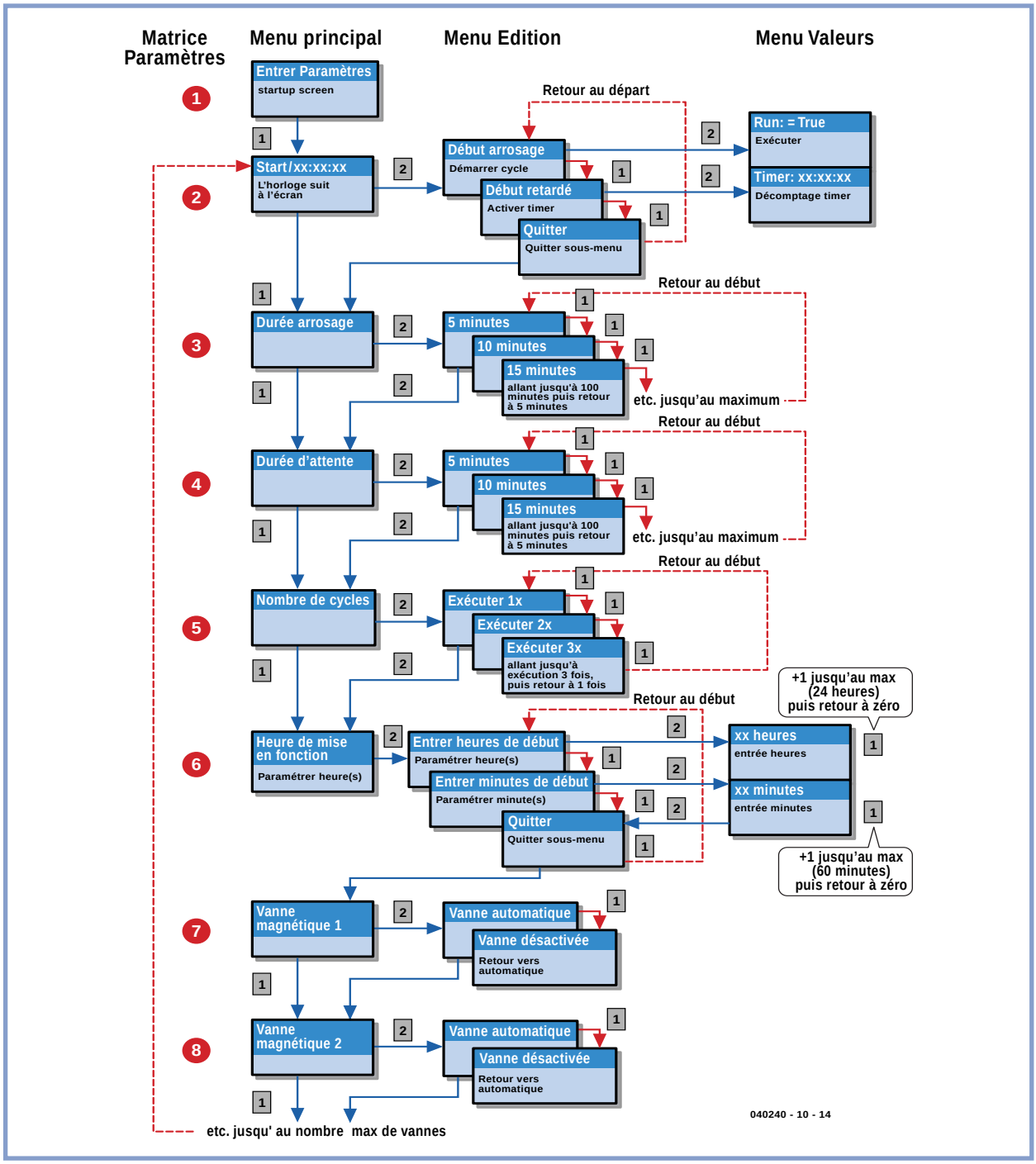


Figure 4. Le menu et l'ensemble de ses fonctions.

Nous y reviendrons.
Lors du traitement du menu dans le sens horizontal (par le bouton 2) on a modification de Menulevel 1. En cas d'ouverture du menu par action sur le bouton 2, on procède à la lecture de la valeur initiale dans la matrice. Celle-ci se trouve à 1 et correspond à l'option de menu « Start direct ». On peut ensuite, par le biais du bouton 1, descendre dans le sous-menu; on voit apparaître, en position 2, l'option de menu « Start delayed » et en 3 l'option « Exit ».
En cas d'une nouvelle action sur le bouton 2, cela se traduira, en fonction de la position (valeur d'edit), sur le schéma, par un saut soit vers la droite soit vers la gauche. Si la valeur edit >2 on a exécution d'un « Exit » et on passe à l'option de menu suivante du menu principal. Si cela n'est pas le cas, on aura, si editmode=1, démarrage du système par mise à True de la variable Run. Si editmode=2 c'est la variable Timer qui est mise à True. Ces variables définissent ce que fera le programme lorsqu'il en sera arrivé à la procédure ControlExe.
Il nous faut d'abord en terminer avec le menu. Le principe est le même pour le paramétrage du timer, à ceci près qu'une seconde action sur le bouton 2 se traduit par le passage au niveau suivant vers la droite, vers Value-mode. C'est la procédure « ShowTimer » qui se charge des textes à visualiser sur l'écran.
En value-mode, on utilise une action sur le bouton 1 pour incrémenter la valeur. C'est de la valeur de edit-mode que dépend l'incrémentation soit de S_hour soit de S_min. Une 3ème action sur le bouton 2 nous fait revenir un niveau en arrière (vers la gauche) dans le menu de sélection.
En fin de menu de sélection on a la possibilité, par « Exit », de quitter le menu de sélection et de revenir au menu principal.
Un mot encore au sujet des variables S_hour et S_min. Elles fixent ensemble le moment où nous voulons lancer le cycle que nous avons paramétré. En face nous avons l'heure actuelle. Elle est suivie par le biais des variables T_hour, T_min et T_sec. Le DelphiStamp ne possède pas d'horloge en temps réel, raison pour laquelle nous utilisons un compteur qui ne cesse d'incrémenter ces variables. La mise à l'heure de l'horloge se fait depuis le PC. Nous en arrivons au point suivant. Pour ce faire, il faut que l'on puisse, depuis le PC, écrire dans la mémoire du DelphiStamp. Pas de problème de ce côté-là, grâce à M485.DLL. Cette DLL s'adresse au serveur 485 présent dans le DelphiStamp. L'adjonction de 3 lignes dans l'unité Umain du projet DelphiStamp (et donc pas dans le projet de simulation !) permet d'activer le serveur.
Le point suivant est l'emplacement de mémoire que nous voulons lire ou où nous voulons écrire. Par la mise en place des variables dans une unité spéciale que nous appellons en premier depuis Umain du DelphiStamp, celles-ci sont placées aux premiers emplacement de la mémoire. Un examen de la cartographie du DelphiStamp nous montre que le BIOS se trouve en début de mémoire, allant jusqu'à $17F. La zone de mémoire que nous pouvons utiliser démarre donc à $180. Nous savons ainsi à quel endroit nous pourrons trouver notre première variable à lire ou à écrire. Il suffit ensuite de compter.
Nous allons utiliser à cet effet une unité UMem.

Nous y plaçons les variables suivantes :
T_hour, **T_min**, **T_sec** du type Byte, pour réaliser l'horloge.
S_hour, **S_min** du type Byte, pour définir le moment de début d'opération.

Depuis UControl nous y déplaçons :
Settings : Array[1..10] of Byte, pour pouvoir lire et écrire les paramètres.
Run du type Byte pour pouvoir déterminer l'état. (0 = False, >1 = True).
Nous ajoutons **Timer** du type Byte pour pouvoir déterminer l'état de la fonction Timer.
C'est tout.

Passons à Ucontrol

Nous incluons l'unité UMem dans la partie Uses de Ucontrol pour pouvoir l'utiliser.
Nous ajoutons, dans la procédure ControlInit, l'initialisation du bus I2C (TWI_init) et les nouvelles variables et nous définissions le paramétrage de base du menu en remplissant la matrice des valeurs par défaut.

C'est dans la procédure ControlExe que se font les plus grandes modifications.
Nous commençons par ajouter un morceau de code destiné à suivre l'horloge. On procède à un appel de la procédure ControlExe toutes les 100 ms. Il nous faut donc compter à 10 pour avoir une seconde, puis à 60 pour les minutes et à 24 pour les heures. Lorsque nous en sommes arrivés à 24 heures tout est remis à zéro pour démarrer le jour suivant.
Cette horloge est visualisée sur la ligne de base du menu principal. Pour cela, nous adaptons le code chargé de montrer la ligne de base. Une fois la fonction timer activée, la ligne de base est modifiée pour afficher le message « Timer » suivi du décompte. À cet effet, nous avons ajouté un morceau de code qui calcule le décompte à partir de T_hour/T_min/S_hour/S_min.
Pour les changements dans le menu, on modifie la section chargée du traitement d'EditMode. Nous ajoutons à menulevel 1 l'appel de ShowStart, pour menulevel 5 on remplace ShowValve par ShowTimer, ShowValve étant transféré vers menulevel > 5.
Lorsqu'arrive le moment de début d'opération, nous faisons passer Run à True. L'instant d'activation de Run est le moment où nous procédons à la lecture des paramètres. Il reste possible, jusqu'à cet ultime instant, de les modifier (de l'extérieur). Nous avons ajouté un morceau de code pour effectuer la lecture des paramètres lors de l'activation du mode Run. Ceci se passe quelques instants avant le lancement du cycle.
Venons-en aux E/S. La prise en compte des boutons-poussoirs est remplacée par la lecture du circuit I2C. De cette façon nous déterminons le bouton actionné et la procédure à appeler alors.
Il reste encore à adapter la procédure SetValve au circuit I2C. Pour cela on écrit, en fonction de la vanne souhaitée, à l'aide de PCF8574_Wr, une valeur de bit vers le circuit. En même temps que la vanne on a également activation du bit de la pompe. À la fin d'un cycle tout est désactivé.
Nous avons terminé toutes les modifications du programme.

Lorsque, dans l'option de menu « Start » nous appuyons sur le bouton 2, nous arrivons, dans le sous-menu, à l'option « Start sprinkling ». Une nouvelle action sur le bouton 2 démarre le cycle.
Nous voyons la LED la plus à gauche s'activer et la première vanne activée, normalement la LED la plus à droite de la vanne 1 (à condition de ne pas avoir modifié de paramètre et que la vanne 1 soit désactivée). Une fois la durée d'arrosage écoulée, on passe à la vanne suivante. 1A la fin du cycle, la désactivation de la dernière vanne entraîne également la désactivation de la pompe.

Dernier point à aborder, le démarrage du timer.
Si nous allons dans l'option de menu « Starting time » du menu principal et que nous appuyons sur le bouton 2 nous arrivons dans le sous-menu et à l'option « Set start hour ». Une nouvelle action sur le bouton 2 nous fait descendre d'un cran dans les sous-menus où nous pouvons entrer l'heure correspondant au moment de déclenchement de l'arrosage par le biais du bouton 1. Ceci fait, nous pouvons revenir au sous-menu précédent par le bouton 2. Par le biais du bouton 1 nous descendons d'un cran dans le menu vers l'option « Set start minute ».
Après avoir entré les minutes à la valeur requise, nous revenons en arrière à l'aide du bouton 2. À l'aide du bouton 1 nous poursuivons vers l'option de menu « Exit » pour revenir ensuite au menu principal et à l'option suivante à l'aide du bouton 2.
L'heure est fixée. Si nous allons à l'option de menu « Start » et que nous appuyons sur le bouton 2, vous pouvons, par le biais du bouton 1, choisir l'option « Delayed start ». Une nouvelle action sur le bouton 2 l'active, ce qui se traduit par l'affichage à l'écran du message « Timer xx:xx:xx », qui indique le temps restant avant déclenchement du cycle. Une nouvelle action sur le bouton 2 et cette fonction est désactivée.
Lors de l'activation du timer, on a, en interne, mise de la variable Timer à True. À chaque fois, on a, dans ControlExe, comparaison entre l'heure et le moment de début d'opération puis réactualisation de l'écran.
Lorsque le résultat de la comparaison est nul, Run est mis à True pour déclencher le cycle, la variable Timer étant elle mise à False. Une fois l'opération terminée, le système se désactive.

Attention : Si l'on vient juste d'initialiser toutes les variables de temps, il peut arriver que l'on ait une activation immédiate lors de l'appel de la fonction de timer. La comparaison repose sur une comparaison des heures et minutes dans T_hour/S_hour et T_min/S_min. En cas d'activation de la fonction timer au cours de la première minute qui suit l'initialisation du DelphiStamp ces variables se trouvent encore toutes à 0, ce qui se traduit par un enclenchement.

Venons-en au DelphiStamp

L'unité UControl créée et testée sous Delphi est recopiée dans le répertoire où a été placé le projet pour le compilateur AVR. L'unité UMem est nouvelle elle aussi, on la copiera également.
Nous lançons ensuite AVRpas et créons un nouveau projet. Veillez à un paramétrage correct (cf. ce qui a été dit à ce sujet dans l'article du mois dernier) et baptisez le projet. Ajoutez le fichier UMain et faites-en le fichier main.
Ouvrez ensuite les autres fichiers dans l'éditeur par File/Open et ajoutez UDrivers.pas, UControl.pas et l'unité UMem. Pour la commande par I²C, il a été réalisé une extension pour l'unité UDrivers. C'est à dessein que nous l'avons mis à part de manière montrer qu'il est extrêmement simple de créer des modules que l'on pourra utiliser lorsque l'on en a besoin. Cette extension s'appelle UDriversEx et elle est inclue dans les uses de UControl.
Compilez ces 2 fichiers par sélection de l'onglet et suite Compile/Current editor file. Il peut arriver qu'il y ait des messages d'erreur, AVRpas ne connaissant pas certaines fonctions Delphi. Il est souvent possible de les contourner par l'une ou l'autre adaptation simple de sorte que l'on peut quand même arriver à une compilation sans erreur.

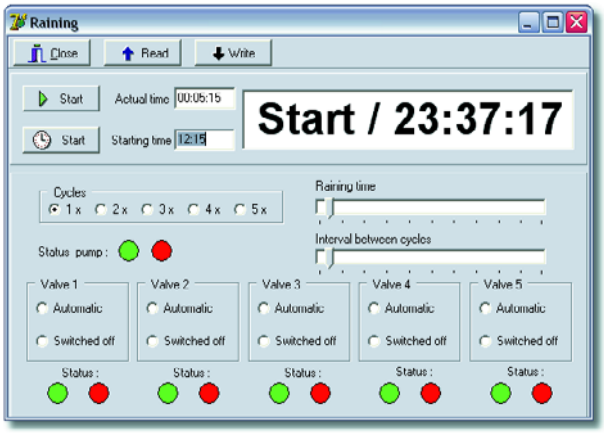


Figure 5. Le programme de suivi (moniteur) et de configuration.

Compilez le projet : Project/Main Project file. Une fois le code généré, on pourra quitter AVRpas.

Le fichier créé par le compilateur croisé, c'est lui qui contient le code binaire, doit être transféré dans le DelphiStamp. Nous utilisons pour cela, par le biais de la liaison RS-232, le programme Mon485 qui nous permet d'entrer en contact avec le DelphiStamp et le M485-server qui y a été démarré. Procédez au transfert du programme; le DelphiStamp est devenu un automate programmable autonome.

Un PC pour le suivi

Le suivi des opérations (*monitoring*) sur l'ordinateur se fait par le biais d'un programme distinct. Par suivi nous entendons la possibilité qu'a le programme de lire toutes les fonctionnalités et de les afficher.
Il est même possible, dans ce cas-là, par l'intermédiaire du bouton Write, d'envoyer vers le DelphiStamp les nouveaux paramètres à définir.
Normalement, le programme se trouve en mode lecture (Read). Nous retrouvons tous les composants dans le programme de moniteur : l'affichage LCD, les paramètres, les 2 options de départ et les actions à exécuter. À cela s'ajoutent l'heure actuelle fournie par le PC et les boutons permettant la lecture ou l'écriture vers le DelphiStamp des paramètres. Ce programme communique avec le DelphiStamp par le biais du protocole M485 et est en mesure d'interroger les paramètres qui y sont stockés.
Les boutons « Start » et « Timer Start » nous permettent de faire passer le DelphiStamp en mode « Run » ou « Timer ».
Avec la description de cette application pour le DelphiStamp nous en sommes arrivé à la fin de ce cours de « Delphi pour les électroniciens ». Nous espérons qu'il vous aura appris quelque chose.

Pour de plus amples informations concernant le DelphiStamp vous pouvez faire un tour à l'adresse www.vogelaar-electronics.com. Nous avons le regret de devoir vous informer que suite aux modifications contractuelles imposées par Borland pour sa nouvelle version de Delphi, le site www.learningdelphi.info a du changer son fusil d'épaule. Sachez cependant que le logiciel **Lazarus** proposé à cet endroit permet également de suivre le cours Dephi dont vous trouvez ici la dernière partie.

(040240-10)