ΑΝΑΦΟΡΑ ΕΡΓΑΣΙΑΣ 3 - Μικροεπεξεργαστές και Περιφερειακά

Στην Τρίτη εργασία κληθήκαμε να δημιουργήσουμε ένα έξυπνο θερμόμετρο. Το θερμόμετρο θα έχει τη δυνατότητα να ανιχνεύει το αν καποιος το προσέγγισε ικανοποιητικα οπότε και θα εμφανίζει πληροφορίες για την θερμοκρασία σε μια οθόνη LCD, ενω ανα 2 λεπτά θα εμφανίζει το μέσο όρο των 24 μετρήσεων θερμοκρασίας που πάρθηκαν ανα 5 δευτερόλεπτα τα προηγούμενα 2 λεπτά. Επίσης, εαν η θερμοκρασία υπερβεί μια συγκεκριμένη τιμη ή πέσει κάτω απο μία αλλη τιμή στην οθόνη θα εμφανίζονται αντίστοιχα μηνύματα και θα ανάβουν κάποια LEDs.

ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ

Στο αρχείο main.c, αρχικά εισάγουμε όλα τα απαραίτητα .h αρχεία, κάνουμε define τα Pin του board στα οποία θα συνδεθούν ο αισθητήρας εγγύτητας(Echo_pin = PA_0,Trigger_Pin = PA_1) και ο αισθητήρας θερμοκρασίας(Temperature_Sensor_Pin = PC_1), ορίζουμε κάποιες global volatile μεταβλητές που θα αλλάζουν τιμές εντός των interrupt handlers που θα ακολουθήσουν και θα περιγραφούν αναλυτικά, καθώς επίσης και κάποιες απλές global μεταβλητές που βοήθησαν στην υλοποίηση του θερμομέτρου όπως τα όρια θερμοκρασίας πάνω και κάτω απο τα οποία ανάβουν τα αντίστοιχα LEDs (όπως ορίζει η εκφώνηση), ένας πίνακας στον οποίο αποθηκεύονται οι 24 μετρήσεις θερμοκρασίας που λαμβάνονται ανα 5 δευτερόλεπτα για 2 λεπτά, και άλλα χρήσιμα flags. Αξίζει να αναφερθεί ότι γίνεται και ο ορισμός 2 μεταβλητών τύπου TIM_HandleTypeDef (struct) των TIM3_Handle και TIM4_Handle που βοηθούν στην αρχικοποίηση των timer 3 και 4 που χρησιμοποιήθηκαν πέρα του Systick timer που περιελάμβαναν οι drivers που δόθηκαν.

Στην συνάρτηση Initializations() που ακολουθεί γίνονται τα περισσότερα initializations που αφορούν τα leds, την LCD με κλήση των αντίστοιχων συναρτήσεων , καθώς επίσης και η αρχικοποίηση των pin του αισθητήρα εγγύτητας όπου θέτουμε το Echo_Pin σε PullDown αντίσταση ,το Trigger_Pin σε Output mode όπως υποδεικνύει το Datasheet του αισθητήρα ενω στο Trigger_Pin δίνουμε την τιμή 0.

Στην ίδια συνάρτηση γίνεται η αρχικοποίηση των δύο extra timers που θα χρησιμοποιηθούν. Η αρχικοποίηση αυτή γίνεται με μία σειρά εντολών που αλλάζουν κάποιες απο τις εσωτερικές μεταβλητές που περιλαμβάνει το struct TIM_HandleTypeDef TIM3_Handle που δημιουργήσαμε παραπάνω. Για τον timer 3 αρχικα τον ενεργοποιούμε και στην συνέχεια συμπληρώνουμε τα μέλη της δομής που προαναφέρθηκε. Έτσι, αρχικά θέτουμε στον Prescaler την τιμή 1599 καθώς το αρχικό ρολόι του timer είναι στα 16Mhz και εμείς επιθυμούμε να το ρίξουμε στα 10Khz, έπειτα θέτουμε στην μεταβλητή CounterMode την τιμή TIM_COUNTERMODE_UP ώστε ο timer να μετρά προς τα πάνω, και θέτουμε την περίοδο του timer σε 1900. Αυτο θα αναγκάσει τον timer να μετρήσει απο το 0 μέχρι το 1900 συμφωνα με

το ρολόι των 10Khz επομένως θα αντιστοιχεί σε χρόνο περιόδου 190 ms δηλαδή ο timer θα «σκάει» κάθε 190 ms. Αφού θέσουμε και το instance σε TIM3 η διεύθυνση της μεταβλητή-struct TIM3_Handle περνά σαν όρισμα στις συναρτήσεις HAL_TIM_Base_Init() και HAL_TIM_BASE_Start() ενω καλούμε και τις συναρτήσεις HAL_NVIC_SetPriority και HAL_NVIC_EnableIRQ με την πρώτη να θέτει κάποια priorities για την interrupt line που εξυπηρετεί τα interrupts του timer 3 και τη δεύτερη να ενεργοποιεί τα interrupts για αυτή τη γραμμή. Η ίδια διαδικασία ακολουθείται αμέσως μετά και για τον timer 4 με μόνη διαφορά ότι η περίοδος τίθεται στη τιμή 50000 καθώς με το Clock μετα το Prescaling να είναι στα 10khz κάθε περίοδος θα διαρκεί 5 δευτερόλεπτα.

Ο timer 3 θα χρησιμοποιηθεί στον αισθητήρα απόστασης για αυτό και ο χρόνος που μετρά είναι τα 190 ms, καθώς κάθε 190 ms ο αισθητήρας θα παίρνει μέτρηση ώστε να διαπιστωθεί αν κάποιος προσέγγισε την συσκευή. Αντίστοιχα ο timer 4 ,όπως μαρτυρά ο χρόνος περιόδου του θα χρησιμοποιηθεί ώστε να παίρνονται μετρήσεις θερμοκρασίας κάθε 5 δευτερόλεπτα όπως ζητείται.

Στη συνέχεια, υλοποιούνται οι interrupt handlers των 2 αυτών timer μέσα στους οποίους καλείται η συναρτηση HAL_TIM_IRQHandler με όρισμα τη διεύθυνση των TIM3_Handle και TIM4_Handle, αντίστοιχα η οποία χειρίζεται το interrupt request που ήρθε. Στον handler του timer 3 η μεταβλητή signal_proximity γινεται 1 ενω στον handler του timer 4 η μεταβλητή signal_5secElapsed γίνεται 1.

Temperature Sensor

Παρακάτω, υπάρχουν οι συναρτήσεις Temperature_Sensor_Init(),Temperature_Sensor_Write() και Temperature_Sensor_Read() οι οποίες κατασκευάστηκαν για την αρχικοποίηση και την υλοποίηση του Read/Write απο και πρός τον αισθητήρα θερμοκρασίας, σύμφωνα με το Datasheet του αισθητήρα που δόθηκε στην εκφώνηση. Οι συναρτήσεις αυτές καλούνται μέσα σε μια άλλη συνάρτηση την Temperature_Sensor() όπου και παίρνουμε την μέτρηση του αισθητήρα. Συνοπτικά, αυτό γίνεται με διαδοχικές κλήσεις της συνάρτησης Temperature_Sensor_Write() με κατάλληλα ορίσματα και καθυστερήσεις και μετά με 2 διαδοχικές κλήσεις της συνάρτησης Temperature_Sensor_Read() αποθηκεύουμε σε 2 byte(ένα για κάθε κλήση της Read) την τιμή της θερμοκρασίας, ηοποία τελικά προκύπτει συνενώνοντας τα 2 byte σε μία μεταβλητή για να δημιουργηθεί η τελική τιμή της θερμοκρασίας. Η συνάρτηση Temperature_Sensor() είναι αυτή που θα καλείται κάθε φορά που θα περνούν 5 δευτερόλεπτα(σκάει ο timer) - θα περιγραφει παρακάτω πως θα γίνει αυτό-, οπότε αμέσως μετά την λήψη της μέτρησης της θερμοκρασίας, αυτή αποθηκεύεται σε έναν πινακα 24 θέσεων. Έπειτα, ελέγχουμε αν η μέτρηση που πάρθηκε ξεπερνά το “upper_limit” που έχουμε θέσει(στους 28^{o} C), και αν αυτό ισχύει τότε ανάβουμε ένα κόκκινο και ένα λευκό LED(αντί για διακόπτη) και εμφανίζουμε στην οθόνη αντίστοιχο μήνυμα “It’s Hot”. Αντίθετα, εαν η

θερμοκρασία είναι χαμηλότερη απο το “lowest_limit”(στους 26^{o} C) τότε ανάβουμε ένα μπλέ LED και εμφανίζουμε στην οθόνη το μήνυμα “It’s Cold”.Και τα 3 Leds έχουν τοποθετηθεί στα Pins που χρησιμοποιούν οι drivers που δόθηκαν για τα leds και συγκεκριμένα το κόκκινο στο Pin PA_6 , το άσπρο το PA_5,και το μπλέ στο P_A7. Στην περίπτωση που η μέτρηση της θερμοκρασίας είναι ενδίαμεσα των δύο ορίων τότε απλά αυξάνουμε τον μετρητή του πίνακα που κρατά τις θερμοκρασίες κατα 1(κάτι που γίνεται ακόμα και αν η θερμοκρασία είναι πάνω ή κάτω απο τα όρια που θέσαμε) και σβήνουμε τα Leds που τυχόν ήταν αναμένα απο κάποια προηγούμενη μέτρηση που ήταν εκτός ορίων.

Επιπλέον, υλοποιήθηκε μία συνάρτηση, η Print_Temperature() για τη διευκόλυνση της εμφάνισης της θερμοκρασία στην οθόνη στις διάφορες περιπτώσεις. Έτσι, αυτή η συνάρτηση επιτρέπει μέσω ενός ορίσματος που δίνεται κατα την κλίση να εμφανίσουμε στην οθόνη είτε τη μέση τιμή των 24 μετρήσεων θερμοκρασίας κάθε 2 λεπτά(τιμή ορίσματος = 1), είτε τη μέση τιμή του δίλεπτου που πέρασε και την τιμή της θερμοκρασίας που πάρθηκε τα τελευταία πέντε δευτερόλεπτα στην περίπτωση που κάποιος προσέγγισε τη συσκευή(τιμή ορίσματος= 0).

Η συνάρτηση Temperature_Sensor(), καλείται μέσα απο μία άλλη συνάρτηση, την Temperature_Every5secs(), η οποία με τη σειρά της καλείται απο τη main και συγκεκριμένα μέσα στο while(1) loop που η τελευταία περιέχει. Στην συνάρτηση, Temperature_Every5secs(), ελέγχεται η τιμή αν η τιμή της signal_5secElapsed είναι 1, που σημαίνει ότι ο εκτελέστηκε ο interrupt handler του timer 4 που σκάει κάθε 5 δευτερόλεπτα, επομένως ήρθε η στιγμή να ληφθεί μέτρηση θερμοκρασίας. Εαν λοιπόν είναι 1 αυτή η μεταβλητή, αμέσως γίνεται 0 ώστε να προετιμαστεί για την επόμενη φορά που θα σκάσει ο timer 4, και καλείται η Temperature_Sensor(). Έπειτα, ελέγχεται εαν έχουμε φτάσει τις 24 μετρήσεις κάτι που εαν ισχύει μηδενίζεται ο counter του πίνακα, καθώς και η μεταβλητή που θα αποθηκευτεί ο μέσος όρος και αμέσως μετά υπολογίζεται η μέση τιμή και εκτυπώνεται στην οθόνη. Ο μέσος όρος πρέπει να μείνει στην οθόνη για 10 δευτερόλεπτα και ο τρόπος που συμβαίνει αυτό θα περιγραφεί ξεχωριστά αργότερα.

Proximity Sensor

Για την έναρξη μιας μέτρησης του αισθητήρα απόστασης πρέπει να δοθεί ένας παλμός διάρκειας 10 us στο Trigger_Pin όπως περιγράφεται στο Datasheet του αισθητήρα, για αυτό και υλοποιήθηκε μία συνάρτηση, η Trigger_Pulse() η οποία θέτει το Trigger_Pin στο 1 (μέσω της gpio_set()), έπειτα καθυστερεί για 10 us(μέσω της delay_us()) και τέλος ρίχνει το Trigger_Pin στο 0.

Η λήψη της μέτρησης γίνεται σε μια άλλη συνάρτηση, την Proximity_Sensor(), εκεί αρχικά ελέγχεται εαν η τιμή της μεταβλητής signal_proximity είναι 1 που σημαίνει ότι εκτελέστηκε ο interrupt handler του timer 3, οπότε πέρασαν 190 ms και ήρθε η στιγμή να ληφθεί μέτρηση

απόστασης. Εάν, λοιπόν είναι 1 η μεταβλητή τότε αυτή μηδενίζεται για να ετοιμαστεί για την επόμενη φορά που θα σκάσει ο timer 3 και καλείται η Trigger_Pulse(), ώστε να δοθεί ο εναρκτήριος παλμός.

Σύμφωνα με το Datasheet το αισθητήρα, για να μετρηθεί η απόσταση πρέπει να μετρηθεί το πλάτος του παλμού που δημιουργήθηκε στο Echo_Pin, για πόσο χρόνο δηλαδή έμεινε στο 1 το Echo_Pin πρωτού πέσει ξανά στο 0, απο όπου και ξεκίνησε μιάς και το έχουμε αρχικοποιήσει σε PullDown resistor. Για τον υπολογισμό αυτού του χρόνου η βέλτιστη λύση θα ήταν να υλοποιήσουμε έναν input capture timer που θα ξεκινούσε να μετρά από όταν το Echo_Pin γινόταν 1 και θα σταμάταγε όταν το Echo_Pin γινόταν 0. Αντί αυτού η μέτρηση του χρόνου υλοποιήθηκε με polling . Ετσι, μία while τρέχει και δεν κάνει τίποτα όσο το Echo_Pin είναι low, μόλις γίνει high ,βγαίνουμε απο τη while αυτή, και κρατάμε σε μία μεταβλητή το την τιμή του counter του timer 3. Αμέσως μετά ακολουθεί μία άλλη while η οποία δεν κάνει τίποτα όσο το Echo_Pin είναι high, ενω μόλις γίνει Low βγαίνουμε απο τη while αυτή και αποθηκεύουμε την τιμή του Counter του timer 3 σε μία άλλη μεταβλητή.Ακόμα και με το polling η ακριβεια που πετυχαίνουμε είναι πολυ ικανοποιητική, ενω λόγω του ότι είναι πολύ μικρός ο χρόνος που καθυστερούμε το πρόγραμμα, δεν επιρεάζεται η υπόλοιπη λειτουργία του προγράμματος.

Αφαιρώντας τις τιμές των 2 μεταβλητών που κράτησαν τις τιμές του counter (όταν το Echo_Pin έγινε high και αντίστοιχα όταν έγινε low) και πολλαπλασιάζοντας με 100 βρίσκουμε το χρόνο σε us που έμεινε high το Echo_Pin(clock timer = 10Khz, T-cycle = 0.1ms) και με κατάλληλη μετατροπή όπως υποδεικνύει το manual του αισθητήρα την τελική απόσταση σε cm. Εαν η απόσταση που υπολογίστηκε είναι μικρότερη απο μία τιμή που θέτουμε εμείς- 10 εκατοστά-, τότε εκτυπώνεται στην οθόνη ο μέσος όρος του προηγούμενου διλέπτου και η τελευταία τιμή θερμοκρασία που λήφθηκε, με κλήση της βοηθητικής συνάρτησης Print_Temperature με όρισμα 0 και μένει στην οθόνη μέχρι να παρθεί η επόμενη μέτρηση θερμοκρασίας, δηλαδή το πολύ 5 δευτερόλεπτα περίπου.

Main Function

Μέσα στη main αρχικά καλείται η συνάρτηση Initializations() που περιγράφηκε παραπάνω, και έπειτα αρχικοποιείται ο systick timer. Με κλήση της timer_init() με όρισμα 1000000 θέτουμε την περίοδο του timer στο 1 second και με τη συνάτηση timer_set_callback(timer_isr) ορίζουμε ποιά συνάρτηση θα καλείται κάθε φορά που περνά ένα δευτερόλεπτο και ο interrupt handler του systick timer «σκαει». Ο timer αυτος χρησιμοποιείται για την υλοποίηση της εμφάνισης του μέσου όρου στην οθόνη για 10 δευτερόλεπτα. Μέσα στον interrupt handler του systick, που ονομάσαμε timer_isr(), αυξάνεται μια μεταβλητή η signal_10secElapsed κάθε φορά που σκάει ο timer. Επίσης μέσα στον handler ελέγχεται εαν η μεταβλητή αυτή έχει φτάσει την τιμή 10, οπότε και τίθεται μια άλλη μεταβλητή , η timer_systick_flag σε 1 ώστε να ειδοποιηθούμε ότι τα 10 δευτερόλεπτα πέρασαν, και η signal_10secElapsed μηδενίζεται.

Επανερχόμενοι στη main, στη συνέχεια ενεργοποιούμε τα interrupts και θέτουμε όλα τα σήματα που αλλάζουν μέσα στους handlers στο 0- δηλαδή στην αρχική τους τιμή, διότι πολλές φορές η ενεργοποίηση των interrupts ενεργοποιεί και καλεί και τους handlers χωρίς να το επιθυμούμε. Ακολουθεί η ατέρμονη while(1) μέσα στην οποία πρώτα καλείται η Temperature_Sensor() και μετά η Proximity_Sensor(). Ο υπολοιπος κώδικας στη main εξυπηρετεί την εμφάνιση για 10 δευτερόλεπτα του μέσου όρου στην οθόνη και θα εξηγηθεί αμέσως μετά.

Μέσος όρος τιμών θερμοκρασίας για 10 δεύτερα στην οθόνη

Αρχικά ,όπως είδαμε ο μέσος όρος θα τυπωθεί πρώτη φορά ,αφού υπολογιστεί και προφανώς αφού έχουν περάσει 24 μετρήσεις, μέσα στην συνάρτηση Temperature_Every5secs(). Επειδή, όμως είναι πιθανό σε εκείνα τα 10 δευτερόλεπτα που πρέπει να φαίνεται στην οθόνη ο μέσος όρος, κάποιος να πλησιάσει τη συσκευή (οπότε και πρέπει να εμφανιστεί στην οθόνη ο μέσος όρος μαζί με την τελευταία τιμή θερμοκρασίας που έχει μετρηθεί) ορίσαμε διάφορες μεταβλητές – flags για να υλοποιήσουμε την επιθυμητή λειτουργικότητα.

Σκοπός μας είναι να καταφέρουμε κατα τη διάρκεια των 10 δευτερολέπτων που εμφανίζεται ο μέσος όρος, να δίνεται η δυνατότητα στον αισθητήρα απόστασης να εμφανίζει το μήνυμα που πρέπει εαν κάποιος προσέγγισε τον αισθητήρα στο παράθυρο των 10 δευτερολέπτων του μέσου όρου. Το μήνυμα του αισθητήρα απόστασης μένει μόνο όσο η απόσταση μας απο τον αισθητήρα είναι μικρότερη απο την προβλεπόμενη. Όταν απομακρυνθούμε και εαν δεν έχουν περάσει 10 δευτερόλεπτα απο την πρώτη εμφάνιση του μηνύματος του μέσου όρου, ο μέσος όρος εμφανίζεται και πάλι στην οθόνη για όσα δευτερόλεπτα απέμειναν.

Επίσης θέλουμε εκείνα τα 10 δευτερόλεπτα ο μέσος όρος να τυπώνεται στην οθόνη μόνο όταν είναι απαραίτητο. Μόνο όταν δηλαδή παρεμβλήθηκε κάποιο άλλο μήνυμα κατα τη διάρκεια των 10 δευτερολέπτων και όχι συνεχώς, ώστε να βλέπουμε ξεκάθαρα το μήνυμα στην οθόνη χωρίς να τρεμοπαίζει(πράγμα που οφείλεται στην συνεχη διαδοχή εκτύπωσης και καθαρισμού της οθόνης).

Για να πετύχουμε τα παραπάνω μέσα στη συνάρτηση Temperature_Every5secs()όταν περάσουν 24 μετρήσεις ,θέτουμε μια μεταβήτή, την calculating_mean ίση με 1, ενεργοποιούμε τον systick timer, και τυπώνουμε τον μέσο όρο στην οθόνη.

Επίπλεον, εαν έχει παρεμβληθεί μήνυμα απο τον αισθητήρα απόστασης, μια μεταβλητή , η print_mean , παίρνει την τιμή 1, ώστε ο μέσος όρος να εκτυπωθεί στην οθόνη και πάλι.

Μέσα στη main, και προτού βγούμε απο τη while(1), ελέγχουμε εαν η μεταβλητή calculating_mean έχει την τιμή 1 και η μεταβλητή print_mean έχει επίσης την τιμή 1. Τότε σημαίνει οτι κάτι παρεμβλήθηκε προτού περάσουν 10 δευτερόλεπτα και έτσι εκτυπώνουμε και πάλι τον μέσο όρο στην οθόνη.Έπειτα ελέγχουμε εαν έχουν παρελθει τα 10 δευτερόλεπτα ,κοιτώντας την τιμή του timer_systick_flag. Εαν η τιμή του είναι 1, πέρασαν 10 δευτερόλεπτα, οπότε θέτουμε την τιμή της μεταβλητή αυτής στο 0(προετιμασία για την επόμενη φορά που θα παρέλθουν 10 δευτερόλεπτα) καθώς επίσης και την τιμή της calculating_mean στο 0 ,και απενεργοποιούμε τον systick timer(Εικόνα 1).

```
if(calculating_mean){

        if(print_mean){
                        Print_Temperature(mean_temperature, 1);
                        print_mean = 0;//reset to 0,so if nothi
        }

        //check if 10 secs elapsed. timer_systick_flag changes t
        if(timer_systick_flag){
                //reset to their first values all variables that
                        print_mean = 1;
                        calculating_mean = 0;
                        timer_systick_flag = 0;
                //disable timer and clear Lcd
                  timer_disable();
                        lcd_clear();
                        lcd_set_cursor(0,0);
```

Εικόνα 1: Ελεγχος για επανεμφάνιση του μεσου όρου

Drivers LCD screen

Αναφορικά με τους Drivers που δόθηκαν για την οθόνη Lcd, υπήρχαν κάποια προβλήματα σε διάφορα σημεία τα οποία φαίνεται να επιλύθηκαν με τις διορθώσεις που προτάθηκαν απο τους συναδέλφους στο forum του μαθήματος. Στην παρακάτω φωτογραφία που ακολουθεί παραθέτω το screenshot της διόρθωσης όπως ανέβηκε αυτούσιο στο forum.

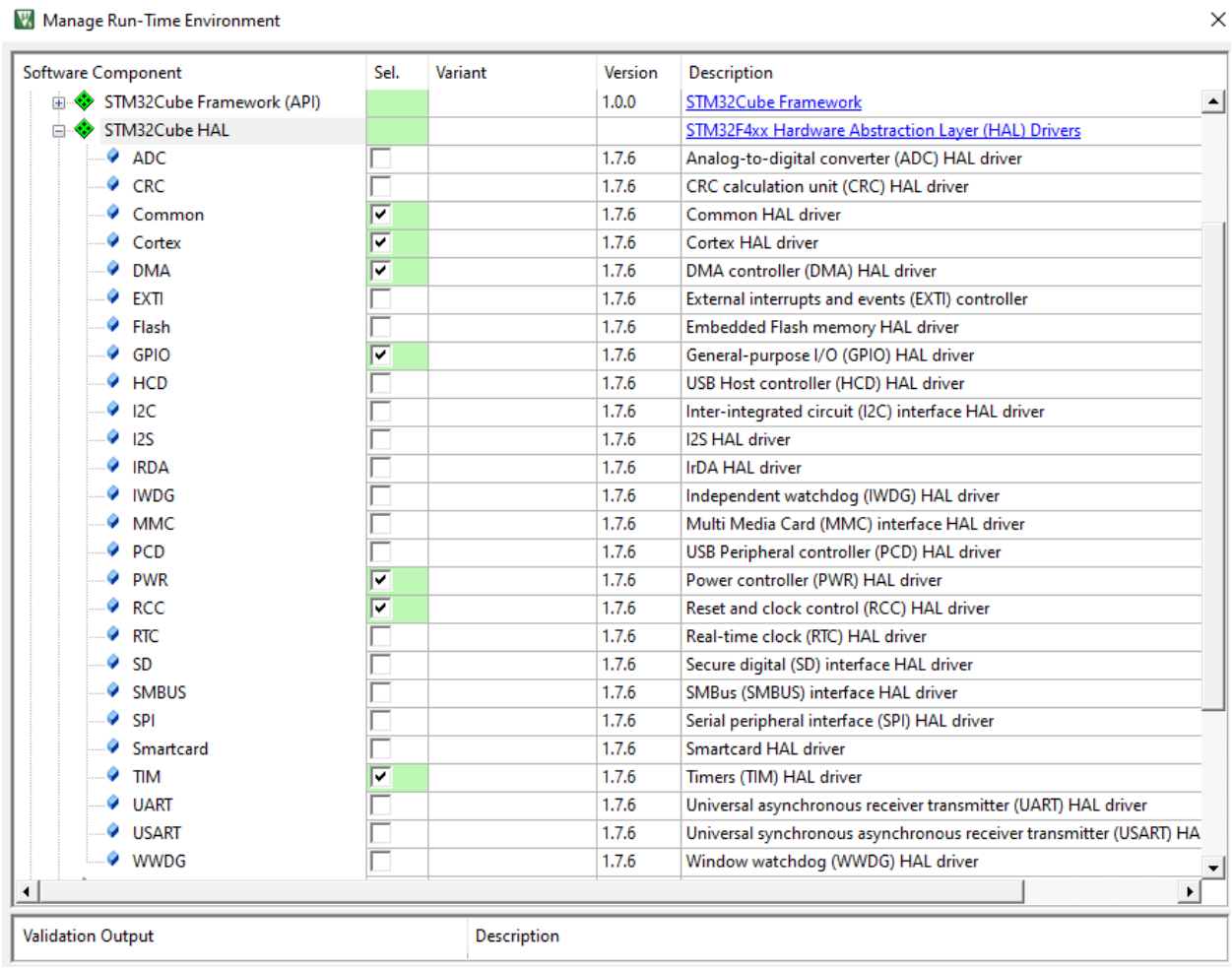
Screenshot 2

```
uint8_t lcd_read_status(void) {
  uint8_t status;
  int i;

  set_data_dir(Input);
  gpio_set(P_LCD_RS, 0);
  gpio_set(P_LCD_RW, 1);
  delay_us(1);
  gpio_set(P_LCD_E, 1);
  delay_us(1);
  //status = gpio_get_range(P_LCD_DATA, 4) << 4;
  for(i=0; i<4; i++) {
    status = gpio_get(P_LCD_DATA[i]) << (4-i);
  }
  gpio_set(P_LCD_E, 0);
  delay_us(1);
  gpio_set(P_LCD_E, 1);
  delay_us(1);
  //status |= gpio_get_range(P_LCD_DATA, 4);
  for(i=0; i<4; i++) {
    status |= gpio_get(P_LCD_DATA[i]) << (8-i);
  }
  gpio_set(P_LCD_E, 0);
  set_data_dir(Output);
  return status;
}
```

Προσθήκη επιπλέον Drivers

Όπως είδαμε, για την υλοποίηση της εργασίας χρησιμοποιήθηκαν και άλλοι timers, πέρα απο τον Systick timer που δινόταν. Αυτό έγινε ώστε να είναι περισσότερο ξεκάθαρα τα χρονικά διαστήματα που μετρά ο καθένας τους χωρίς να χρειάζεται να εχουμε πολλούς counters μέσα στον handler του systick , και ενδεχόμενες αλλαγές να γίνονται γρηγορότερα. Για την εισαγωγή των επιπλέον drivers έγιναν οι παρακάτω προσθήκες που φαίνονται στη φωτογραφία απο το “Manage Run-time Environment” του keil.



Testing

Για να διαπιστωθεί η καλή λειτουργία του προγράμματος, αρχικά, ελέγχθηκαν ξεχωριστά τα περιφερειακά (εσωτερικά και εξωτερικά). Έτσι, χρονομετρήθηκαν οι χρόνοι που θέλαμε να σκάσουν οι timers για την λήψη θερμοκρασίας κάθε 5 δευτερόλεπτα αλλα και για την εμφάνιση του μέσου όρου μετά το πέρας των 2 λεπτών και την εμφάνιση του για 10 δεύτερα στην οθόνη. Επίσης, με τη βοήθεια ενός χάρακα διαπιστώθηκε ότι ο αισθητήρας εγγύτητας δουλεύει με μεγάλη ακρίβεια ενώ με χρήση θερμομέτρου εσωτερικού χώρου τσεκάραμε και τις τιμές του αισθητήρα θερμότητας.

Για το κομμάτι που αφορά το άναμμα των Leds και τα μηνύματα στην οθόνη όταν η θερμοκρασία βγεί εκτός άνω και κάτω ορίου, αλλάξαμε τις τιμές upper_limit και lower_limit ώστε να «αναγκάσουμε» τη θερμοκρασία δωματίου να βγεί εκτός ορίων και να διαπιστώσουμε ότι γίνονται οι σωστές ενέργειες σε κάθε περίπτωση.

Στο video που συμπεριλαμβάνεται, καλυπτονται όλες οι περιπτώσεις και διαπιστώνεται η καλή λειτουργία και της οθόνης που εμφανίζει τα κατάλληλα μηνύματα κάθε φορά για όσο χρόνο θέλουμε.

Σύντομη επεξήγηση Video

Στο βίντεο που συμπεριλαμβάνεται αρχικά κάνουμε Reset και ανάβουμε όλα τα Leds για να σιγουρευτούμε ότι λειτουργούν. ‘Επειτα(5 δευτερόλεπτα μετά το Reset οπότε και λαμβάνουμε την πρώτη μέτρηση), εμφανίζεται μήνυμα για χαμηλή θερμοκρασία καθώς το όριο κάτω απο το οποίο έχουμε επιλέξει να εμφανίζεται το μήνυμα είναι οι 26^{o} C. Στη συνέχεια, αφου πλησιάζει το χέρι μου σε απόσταση μικρότερη των 10 cm απο τον αισθητήρα εμφανίζεται το κατάλληλο μήνυμα με την θερμοκρασία και το μέσο όρο που ακόμα δεν έχει υπολογιστεί οπότε και έχει την τιμή 0.00. Κάποια στιγμή, αγγίζοντας τον αισθητήρα ανεβάζουμε τη θερμοκρασία που λαμβάνει και καθώς αυτή ξεπερνάει τους 28 βαθμούς, παρατηρούμε το κόκκινο και το άσπρο Led(αντι για διακόπτη) να ανάβουν και μήνυμα για άνοδο της θερμοκρασίας να εμφανίζεται στην οθόνη. Μαζί με το μήνυμα για υψηλή ή χαμηλή θερμοκρασία εμφανίζεται και η θερμοκρασία εκείνη τη στιγμή, έτσι με χρήση ενός ανεμιστήρα που κάποια στιγμή ενεργοποιούμε και καθώς η θερμοκρασία είναι χαμηλότερη απο τους 26 βαθμούς(οπότε εμφανίζεται μήνυμα στην οθόνη), παρατηρούμε ότι η τιμή της μέτρησης στην οθόνη καθώς η θερμοκρασία πέφτει, αλλάζει κάθε 5 δευτερόλεπτα πράγμα που επαληθεύει την λειτουργία που ζητήθηκε στην εκφώνηση. Όταν συμπληρωθούν 2 λεπτά εμφανίζεται ο μέσός όρος. Πρίν συμπληρωθούν πλησιάζουμε ξανά τον αισθητήρα και παρατηρούμε οτι εμφανίζει τη τη θερμοκρασία αυτή τη στιγμή καθώς και το μέσο όρο. Μόλις απομακρυνθούμε το μήνυμα αυτό εξαφανίζεται και ο μέσος όρος εμφανίζεται και πάλι όπως προβλέπουμε. Στο επόμενο 2λεπτο πλησιάζουμε τον αισθητήρα απόστασης ανα τακτά χρονικά διαστήματα για

να επιβεβαιωθεί και πάλι η καλή λειτουργία του και με τη συμπλήρωση του 2^{ου} 2λεπτου, βλέπουμε τον μέσο όρο να εμφανίζεται και πάλι με τη διαφορά ότι αυτή τη φορά τον αφήνουμε να εκτυπώνεται αδιάλειπτα για να επαληθευτεί ότι το μήνυμα μένει στην οθόνη για 10 δευτερόλεπτα, όπως και ζητείται

Χρήσιμα links

Τα Links που ακολουθούν βοήθησαν στην υλοποίηση του προγράμματος και συγκεκριμένα στην αρχικοποίηση και στη χρήση των extra timers καθώς και στην υλοποίηση του driver του αισθητήρα θερμοκρασίας

1. https://visualgdb.com/tutorials/arm/stm32/timers/hal/
2. https://arm-stm.blogspot.com/2015/12/timer-interrupt-on-stm32f4-using-hal.html
3. https://www.youtube.com/watch?v=09C1dyXvSbg&feature=youtu.be&t=1140