# Aries ATU (Arduino Nano 33 IoT)

# Ce document s ‘applique à la version la plus récente d'Aries utilisant un processeur Arduino Nano 33 IoT et utilisant un écran Nextion. L'écran est optionnel et offre des affichages de puissance et de TOS si vous le souhaitez. ARIES peut être utilisé de deux façons :

# - En mode normal, il est contrôlé par Thetis pour sélectionner sa fréquence TX et ses antennes TX et RX. TUNE est initié par des commandes CAT directement par Thetis. Un port CAT spécifique est utilisé, car Thetis pousse les informations directement vers Aries.

# - En mode "standalone", Thetis utilise un port CAT normal, et interroge la fréquence TX toutes les quelques secondes. TUNE est initié par une connexion câblée. Ce mode est prévu pour qu'ARIES soit installé dans un amplificateur QRO attaché à une radio Andromeda qui a déjà un ARIES intégré. En mode autonome, Aries n'a aucune connaissance des antennes utilisées.

# Il y a aussi un mode "protection de l'amplificateur" qui est optionnellement inclus dans le code source et qui nécessite un affichage. Ce mode contrôle le "déclenchement" des PTT de l'amplificateur si une condition de défaut (par exemple un VSWR élevé, un excès de puissance directe, une surchauffe...) se produit. Une page d'affichage montre la condition de déclenchement et permet une réinitialisation manuelle une fois que le défaut a été éliminé.

# Concept Aries

## Le concept repose sur l’idée qu'une ATU doive être installée “dans” Andromeda (Andromeda est l’interface matérielle “intégré” supportant le logiciel client Thetis). Kjell a suggéré l'architecture standard L-C.

## Rationale

|  |  |
| --- | --- |
| **Idées de base concernant Aries** | **Raisons** |
| Aries est contrôlé par Thetis à l'aide de commandes CAT. | Interface utilisateur plus précise et affichage de l'état du tuner via Thetis |
| Ariues est connecté à la voie TX de la radio, pas à la voie RX. | Il n'est pas nécessaire de mettre l'ATU hors circuit pour le RX. |
| Aries utilisera l'arrangement classique "L match" avec 8 inductances et 8 condensateurs. | Il fonctionne et les valeurs des composants appropriés sont connues. |
| Aries stocke les solutions d'accord L/C avec une "grille" par pas de 10KHz. | Grille assez fine pour la plupart des antennes (mais peut-être pas les boucles magnétiques) |
| Aries stocke des solutions distinctes pour chacune des 3 sorties d'antenne | Pour que 3 antennes différentes puissent être connectées, avec un réglage séparé pour chacune d'elles. |
| Aries reçoit la fréquence TX correspondant à la commande CAT.  Une nouvelle commande est envoyée chaque fois que la fréquence passe à un nouveau pas de fréquence (~10KHz). | Pour qu'il "connaisse" la fréquence lorsque la radio est réglée et qu'il puisse indiquer immédiatement si une solution mémorisée est disponible ou non.  Cela permet également de ne pas avoir à mesurer la fréquence TX pour un signal SSB, ce qui serait problématique. |
| Aries mesure lui-même le ROS | Il a besoin d'une nouvelle mesure de ROS toutes les 10à16 ms ; Recevoir les informations ROS depuis Thetis pendant le réglage serait bien trop lent. |
| Aries peut être active ou contourné pour chaque antenne individuellement. | Pour permettre sa désactivation si ANT3 est connecté à un linéaire externe, par exemple |
| Il doit y avoir un moyen pour l'utilisateur d’indiquer à Aries d'effacer les solutions de réglage pour n'importe laquelle des 3 antennes, et ceci pour chaque antenne individuellement. | De cette manière, en cas de changement ou de modification d'une antenne, Aries n'essaie pas d'utiliser les "anciennes" solutions. |
| Aries recherche une nouvelle solution de syntonisation lorsque TUNE est sélectionné sur la radio. |  |
| Un encodeur est disponible pour "affiner" les résultats de l'accord de l'ATU (sur la carte de contrôle) | Appui long sur le bouton encodeur : bascule de réglage grossier/fin (taille de pas L/C)  Appui sur le bouton du codeur : bascule entre l'inductance et la capacité. |

## Interface to Thetis

|  |  |
| --- | --- |
| **THETIS action** | **Aries action** |
| Au démarrage :  Trouve la bande et donc l'antenne utilisée.  Vérifie si Aries est activé pour cette bande. Envoie le message “active”/”non active”.  Pour la fréquence TX initiale : envoie la valeur de la fréquence.  Initialise le symbole d'affichage de l'ATU - soit "off" ou "no solution". |  |
| Lorsqu'une antenne est changée, Thetis envoie une commande d'activation ou de contournement à Aries et le nouveau numéro d'antenne.  Si Aries est activé, Thetis affiche un symbole à l'écran  Si une solution de réglage est signalée comme disponible, la LED verte est allumée et le symbole de l'affichage ATU est mis en évidence. | Aries recherche une solution d'accord : sur la fréquence spécifiée, puis jusqu'à ±50KHz d’écart. S'il trouve une solution, il la sélectionne prête à l'emploi ; sinon, il sélectionne "bypass". Notez que les relais ne sont pas modifiés - il se prépare simplement à définir ces valeurs de relais.  Un message est envoyé à Thetis avec la solution disponible ou non. |
| Lorsqu'une bande est changée, Thetis envoie un nouveau message de fréquence à Aries.  Lorsque la radio est accordée de plus de 10KHz par rapport à la dernière fréquence, un message de nouvelle fréquence est envoyé à Aries.  Si une solution d'accord est signalée comme disponible, la LED verte est allumée et le symbole de l'affichage ATU est mis en évidence. | Si elle est activée : Aries recherche une solution d'accord : sur la fréquence spécifiée, puis jusqu'à ±50KHz de distance. S'il trouve une solution, il la sélectionne prête à l'emploi ; sinon, il sélectionne "bypass". Notez que les relais ne sont pas modifiés - il se prépare simplement à définir ces valeurs de relais.  Un message est renvoyé à Thetis avec la solution disponible ou non. |
| Quand TX est activé | Si la fréquence a été modifiée, Aries transmet la nouvelle sélection de relais pour la nouvelle solution d'accord de la mémoire aux relais. Si aucune solution n'était disponible, il sélectionne le contournement.  **Cela doit se faire rapidement – sur activation du PTT.** |
| Lorsque TX est désactivé | Les relais sont laissés dans la même position, prêts pour l'opération suivante. |
| Lorsque TUNE est sélectionné : le voyant rouge est allumé. Un message "tune now" est envoyé à Aries.  Thetis reste dans l'état TUNE et indique le succès ou l'échec à l'aide de la LED verte et du symbole d'affichage en surbrillance. (option permettant de rester dans l'état TUNE pour un réglage fin, ou de sortir simplement). | Aries commence son algorithme pour trouver une nouvelle solution. Une fois terminé, si une bonne solution a été trouvée, elle est stockée dans l'EEPROM.  Aries renvoie un message indiquant "tune complete" et "successful/not successful". |
| Si l'utilisateur demande l'effacement des réglages pour ANT1/2/3  Affiche "effacement..."  Change l'affichage en "Done" (terminé) | Toutes les solutions de réglage pour cette antenne sont effacées de l'EEPROM. Si cette antenne est sélectionnée, Aries entre en état de bypass. (notez que cela prend environ 5 secondes je pense)  ARIES envoie un message de réponse lorsque c'est terminé Si l'antenne sélectionnée == antenne effacée, ARIES envoie "not successful". |



Erasing….Done

H/W 2; S/W 17

CAT port

Com4

Solutions

X

3

2

Erase

Erase

Erase

Ant

ATU enabled

X

X

1

Enabled

f/w version

X

ATU

Figure 1: Onglet de configuration suggéré



ATU

Figure 2: L'affichage d'Andromède ayant un symbole "ATU" :

# Operating Modes

Il existe 2 modes de fonctionnement fondamentaux de l'ATU

Mode Normal

* Aries est installé à l'intérieur d'un émetteur-récepteur (par exemple Andromeda) ou comme un ajout externe.
* Son port CAT doit être réglé sur le port CAT d’Aries dans les paramètres de Thetis Andromeda.
* Aucun affichage Nextion n'est utilisé.

Mode Indépendant (standalone)

Le mode autonome est détecté par un cavalier installé dans J17.

* Dans ce mode, Aries est installé dans un amplificateur de puissance supplémentaire.
* Son port CAT doit être sélectionné parmi les ports CAT1-CAT4 "normaux" de Thetis et NON le port CAT d'Aries.
* Un écran Nextion est utilisé et doit être connecté.
* L'entrée de l'antenne est sélectionnée par des entrées câblées
* TUNE est contrôlé par une entrée câblée à J6.
* Aries interroge la fréquence TX en utilisant une commande CAT.
* Un formulaire de configuration est accessible à partir de la page d'affichage "Engineering". De là, on peut accéder au réglage de l'échelle de puissance maximale du wattmètre et à l'effacement de la solution d'antenne (par exemple si l'antenne est changée). Les résistances de division de puissance avant/arrière doivent être réglées correctement pour l'échelle de puissance prévue.
* La protection de l'amplificateur est sélectionnée automatiquement si un MCP23017 est détecté. Ceci se connecte via J1.   
    
  Les "deltas" du code autonome seront :  
  1. En mode autonome, envoi d’ une commande CAT de demande de fréquence toutes les 5 secondes.  
  2. Prise en compte du changement d'antenne à partir de 2 entrées h/w D2 & D5 (document)  
  3. Prise en compte de l'activation/désactivation de l'ATU à partir de l'écran du Nextion.  
  4. Sauvegarde l'ATU activé/désactivé dans l'EEPROM.  
  5. Renvoie du code pour l'entrée TUNE h/w.  
  6. Concevoir des affichages appropriés  
  7. Sauvegarder l'affichage actuel dans l'EEPROM en mode autonome.  
  8. Sauvegarder l'échelle de l'affichage actuel dans l'EEPROM en mode autonome.  
  9. Pour chaque échelle d'affichage sélectionnée, nous aurons besoin d'un facteur d'échelle ADC pour calculer la puissance.  
  10. Il devra y avoir un moyen d'effacer les paramètres ATU mémorisés pour chaque antenne.  
  11. Connecter une carte de protection de l'amplificateur par une puce I2C supplémentaire pour permettre la surveillance des conditions de défaut dans le PA.  
  12. Après la mise sous tension, vérifier les conditions de défaut et, s'il n'y a pas de défaut, réinitialiser un flip flop sur la carte de protection.  
  13. Prévoir un écran d'affichage supplémentaire en cas de "déclenchement" pour un ROS excessif ou une puissance inverse.  
  14. Bouton RESET sur l'écran d'affichage supplémentaire.  
  15. Affichage des informations sur la bande provenant de la carte de protection.  
  Le mode autonome est détecté par la présence d'un cavalier sur J17.   
  En complément du mode autonome, une fonction de protection de l'amplificateur a été ajoutée. Elle est activée automatiquement en détectant la présence d'un MCP23017 sur le bus I2C.  
    
  ATU et l’abaque de smith

L'abaque de Smith est un tracé du coefficient de réflexion complexe S11 sur un diagramme d'Argand. C'est un moyen utile pour visualiser ce que doit faire un circuit résonant. Sur l’abaque, l’on peut voir :

* - Point central - accord parfait ; impédance de charge = Zo
* Point à gauche - court-circuit, resistance nulle
* Point à droite - circuit ouvert, resistance infinie

|  |  |
| --- | --- |
|  |  |
| Tracé d'impédance complexe | Diagramme d'admittance complexe (image miroir horizontale) |

Il est alors possible d’obtenir une trace des valeurs d'inductance et de capacité, lesquelles déplacent l'impédance d'une charge. Une inductance en série se déplacera dans le sens des aiguilles d'une montre le long d'une ligne de résistance constante. Une capacité shunt se déplacera dans le sens des aiguilles d'une montre le long d'une ligne d'admittance constante.

|  |  |
| --- | --- |
|  |  |

Figure 3: Effets de l'inductance en série et de la capacité en dérivation

En ajoutant L et C de manière appropriée, le réseau d'adaptation L peut s'adapter à n'importe quelle impédance sur le diagramme. Mais la série L, le shunt C ne peuvent réaliser qu'une seule solution, et le fait que le condensateur soit à l'entrée ou à la sortie dépend de l'endroit où l'impédance de charge est placée. Vous pouvez alors tracer les régions où l’ATU en “L” peut s'adapter - les régions grises ne peuvent pas être adaptées :

|  |  |
| --- | --- |
|  |  |
|  |  |

Figure 4: zones d'appariement pour les 2 configuration de l'ATU en “L-Match”

# Réseau d’adaptation

Le réseau d'adaptation est constitué de 8 inductances et de 8 condensateurs dans un agencement classique d'adaptation en L. Si le condensateur est connecté à l'entrée, il s'adapte aux charges à faible impédance ; s'il est connecté à la sortie, il s'adapte aux charges à haute impédance.

1er modèle de test : capacité par pas de 10pF jusqu'à un total de 2800pF ; inductance par pas de 40nH jusqu'à un total de 10uH. Le tuner peut s'adapter à des charges résistives de basse impédance 8:1 et de haute impédance 8:1 à 1.9MHz.

La version 4 de l’ATU a été construite en utilisant les valeurs suivantes :

|  |  |  |  |
| --- | --- | --- | --- |
| Composants | Valeur désirée | **Valeur utilisée (pF)** | Valeur combinée (pF) |
| C33, C34 | 21.1 | 20 | 10 |
| C35, C36 | 42.2 | 39 | 19.5 |
| C38, C38 | 84.4 | 82 | 41 |
| C39, C40 | 168.8 | 180 | 90 |
| C41, C42 | 337.5 | 330 | 165 |
| C43, C44 | 675 | 680 | 340 |
| C45, C56 | 1350 | 1300 | 650 |
| C57, C58 | 2700 | 2700 | 1350 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Inductance souhaitée uH | Tore | Al | Nombre de tours calculés | Nombre de tours bobinés | Inductance obtenue uH |
| 0.023 | 0.3” dia air wound. 1 turn 0.15” length | | | | |
| 0.047 | 0.3” dia air wound. 3 turns 0.2” length | | | | |
| 0.094 | T94-6 | 70 | 3.7 | 4 | 0.11 |
| 0.188 | T94-6 | 70 | 5.2 | 5 | 0.18 |
| 0.375 | T94-2 | 84 | 6.7 | 7 | 0.41 |
| 0.75 | T94-2 | 84 | 9.4 | 9 | 0.68 |
| 1.5 | T94-2 | 84 | 13.4 | 13 | 1.42 |
| 3 | T94-2 | 84 | 18.9 | 19 | 3.03 |

Dans les deux cas : la plus petite valeur sélectionnée par le bit 0 ; le composant est sélectionné en circuit si le bit de données est à 1.

# Le relais de commutation entre le niveau Z bas et le niveau Z haut devra être commandé directement par une broche du processeur.

# messages CAT nécessaires

Les messages CAT existants ont été utilisés dans la mesure du possible

|  |  |  |
| --- | --- | --- |
| **Event** | **Message** |  |
| TX on/off | Par signal hardware |  |
| TUNE on/off | Message CAT ZZTUn;  Émis du PC vers Aries | n=0: no tune; n=1: TUN active |
| Changement de Fréquence | Message CAT ZZFTmmmmmmmmmmm;  Émis du PC vers Aries | mmmmmmmmmmm: 11 digits (fréquence en Hz)  ex 00014320000 = 14.32 MHz  ex 00001850000 = 1.85 MHz  (pas de 10kHz attendus) |
| Changement d’antenne TX | Message CAT ZZOCn;  Émis du PC vers Aries. | n=1: Ant1; n=2: Ant2; n=3: Ant3 |
| Changement d’antenne RX | Message CAT ZZOAn;  Émis du PC vers Aries. | n=1: Ant1; n=2: Ant2; n=3: Ant3 |
| Efface la Solution | Message CAT ZZOZn;  Émis du PC vers Aries.  Réponse: ZZOZn;  Émis d’Aries vers le PC | n=1: efface les solutions pour Ant1; n=2: efface pour Ant2; n=3: efface pour Ant3 |
| Accord fin L/C | Message CAT : ZZZEnnm;  Émis du PC vers Aries. | nn= Nu méro d’encodeur et direction. m= nombre de pas  Valeurs autorisées: 01=L c/w; 02=C c/w; 51=L ac/w; 52=C ac/w |
| ATU success/fail | Message CAT : ZZOXn;  Émis d’Aries vers le PC | n= 0: pas de solution ATU trouvée; n=1: solution acceptable trouvée. |
| ATU active | Message CAT : ZZOVn;  Émis du PC vers Aries | n=0: ATU inactive; n=1: ATU active, accordera sur demande |
| Requête “version du soft” | ZZZS;  Réponse ZZZSppnnmmm; | pp=product id  1: Andromeda 2: Aries 3: Ganymede  nn= version hardware  mmm= s/w version |
| Options Set ATU | ZZOYn; | n =0: déclenche accord complet  n=1: autorise accord rapide (quick tune) |

# Problèmes liés au processeur

Le processeur n'a pas besoin d'être puissant ; il sera inactif la plupart du temps en attendant que les relais se stabilisent. Il doit cependant disposer d'une mémoire suffisante pour contenir les solutions de réglage d'une antenne.

Un Arduino Nano 33 IoT semble convenir. Il possède un processeur SAMD21G18A.

|  |  |
| --- | --- |
|  |  |

## Timers

Les projets précédents suggèrent que je devrais savoir quels timers sont déjà utilisés ! Les fonctions de temps de l'Arduino (eh milis() semblent utiliser le timer "systick" du processeur. D'après le fichier variant.cpp :  
TCC0, TCC1, TCC2, TC3, TC4, TC5 peuvent tous être utilisés pour les sorties PWM. TC5 utilisé pour le PWM sur Dig3  
J'aurai besoin d'une interruption du timer. Il existe une bibliothèque Arduino Zero timer sur Github.com/EHbtj/ZeroTimer qui semble utiliser TC3 et/ou TCC0. La période du timer est spécifiée en microsecondes.

## E/S Hardware

Les relais seront commandés par SPI à l'aide de 3 registres à décallage TPIC6B595.

## Assignation des E/S

|  |  |  |
| --- | --- | --- |
| **Function** | **Pin** | **Comment** |
| Serie | TX/DIG0  RX/DIG1 | Permet l'utilisation de l'écran tactile Nextion I/O |
| SPI | D13/SCK  D11/MOSI | Horloge et données série vers TPIC6B595 |
| I2C | A4 / SDA  A5 / SCL | Pour EEPROM (et MCP23017 sur la carte de protection de l'amplificateur en mode autonome) |
| LOAD | DIG6 | Le front montant charge les données série de SPI |
| Relais haute et basse Z  (Ver. 4 & en deça) | DIG8 | 1=high Z; 0=low Z |
| Détection de déclenchement de la protection de l'amplificateur (Ver. 5) | DIG8 | Entrée du flip-flop S/R.  =1: Normal  =0: déclanché. |
| PTT | DIG10 | 0=TX; 1= RX. Nécessite pullup. Piloté par interruption. |
| Entrée du pont VSWR | A0, A1 | A0=fwd; A1=rev. La tension analogique doit être étalonnée pour 3.3V max |
| Entrée TUNE externe | DIG9 | Nécessite pull-up et interruption. 0=TUNE; 1=no tune. |
| LED de statu | DIG7 |  |
| Sortie T/R (PTT) | DIG3 | =1: PTT asserted (TX) |
| Réglage Encodeur L/C | A2, A3 |  |
| Inter fugitive encodeur | DIG4 |  |
| Entrée antenne (mode autonome) | DIG5, DIG2 |  |
| Sélection Mode autonome | DIG12 | =0 au démarrage : mode autonome  =1: mode normal.  Note : résistance pullup externe nécessaire, cela fait partie des broches SPI, mais cela peut être lu.) |
| Capteur de courant PA | A6 | Entrée analogique |
| Non utilisée | A7 |  |

Entrée de l'antenne en mode autonome : Notez que celles-ci ont besoin d'une diode en série sur le PCB.

|  |  |  |  |
| --- | --- | --- | --- |
| **Antenna** | **Entrée B**  **D5** | **Entrée A**  **D2** | **~D5,~D2** |
| 1 | H | H | 00 |
| 2 | H | L | 01 |
| 3 | L | H | 10 |
| 4 | L | L | 11 |

Changements des broches E/S dans la version 5

* ENC 2 éliminé.
* Cavalier sur D12 pour sélectionner le mode STANDALONE ou NORMAL.
* Sélection Impédance haute/basse déplacée de DIG8 à IC7,D3
* Entrées d'antenne ajoutées sur DIG2, DIG5
* Sortie de réinitialisation de la protection de l'amplificateur sur DIG8
* Moniteur de courant PA sur A6
* A7 est actuellement en reserve

## Assignations I2C

|  |  |
| --- | --- |
| EEPROM | 0x50, 0x51 (includes A16 as bottom bit) |
| MCP23017 (protection mode) | 0x20 (presence auto detected) |

# EEPROM Interface I2C

Une EEPROM externe est nécessaire. Nous avons besoin de 3 octets par fréquence pour stocker les solutions d'accord. Si nous stockons une solution par 10KHz, nous avons besoin de 100 réglages par MHz soit environ 6000 réglages pour la bande HF soit 18 Ko. Si nous avons 3 antennes et des solutions distinctes pour chacune d'elles, cela représente 54Ko soit près de 500Kb. Les EEPROMs de 2Mb+ sont facilement disponibles avec une interface I2C. L'EEPROM 1Mbit de Microchip (24FC1026-I/P) conviendra et une bibliothèque Arduino est disponible. Les dispositifs FC peuvent fonctionner à 1MHz. L'EEPROM se connecte à l'interface I2C.  
  
 Accès des données EEPROM

# Les données pour une antenne entière seront lues dans la SRAM lorsque l'antenne est changée. Il y avait un problème potentiel de “race condition” avec l'EEPROM. Cela peut être entièrement évité maintenant qu'un écran Nextion est utilisé pour le débogage.

# Pour réécrire des solutions individuelles, il peut être utile d'utiliser des écritures d'octets un à un. Pour effacer les paramètres d'une antenne, utilisez l'écriture en bloc (128 octets prennent les mêmes 5ms). Pour lire des données, utilisez la lecture séquentielle avec une transaction d'adresse (~40us) puis des lectures individuelles transférant un seul nouvel octet (10us).

# Pour des raisons de commodité, l'état "sans solution" devrait être la condition expédiée -solution = 0xFFFFFF.

# Chaque bloc de données pour chaque antenne doit commencer à la limite d'une page (128 octets) et être dimensionné de manière à ce qu'un nombre fixe d'écritures de page puisse être effectué pour l'effacer. Cela suggère qu'il devrait être un peu plus grand que nécessaire.

# Il existe une bibliothèque Arduino (extEEPROM) qui supporte la 24FC1026.

# Interface SPI

Les données série seront transférées dans deux registres à décalage TPIC6B595. Un troisième sera prévu pour une carte additionnelle pour piloter un relais T/R et 3 relais de sélection d'antenne : cela permettrait un fonctionnement autonome avec les anciens transceivers HPSDR. L'octet 1 est le premier décalé, atteignant la fin de la chaîne SR.

|  |  |
| --- | --- |
| Byte | Meaning |
| 1 | Bit 0: sélectionne l’antenne 1. 0 = antenne sélectionnée.  Bits 2:1: sélectionne l’antenne 3-2. 1 = antenne sélectionnée.  Avec le PCB V 5.xx: Bit 3 = High/LowZ. 1= High Z |
| 2 | Valeur Capacitance. Bit 0 = le plus faible. |
| 3 | Valeur Inductance. Bit 0 = le plus faible. |

Les réglages des relais RX et TX peuvent être différents. Dans la plupart des cas, ce registre sera écrit à chaque fois qu'il y aura un changement de T/R. Cependant, tout changement d'antenne RX doit être envoyé au matériel immédiatement, à moins que TX ne soit déjà en cours.

Les données SPI doivent être écrites chaque fois que PTT change d'état, quand l'antenne RX change, ou quand une nouvelle solution TUNE est envoyée pendant le réglage. Il y a 2 “race condition” potentielles Si les données de l'antenne RX sont déjà en cours d'écriture lorsque le PTT est activé, les données devront être réécrites à la fin du décalage. Un drapeau devra être activé.

# Si le strobe TX est désactivé pendant l'écriture de la solution d'accord pendant l'algorithme “tune”

# PTT

Lorsque vous appuyez sur le PTT, vous obtenez plusieurs événements de "pression". Cela pourrait être dû à la présence de HF, faisant varier GND. Il est preferable d’utiliser : “Set PTT from interrupt”;

1. Régler le timer “min PTT duration” à 32 ms;
2. Sonder le PTT pour voir s'il doit être relâché.
3. Lorsqu'il est relâché, le minuteur "min PTT inactif" doit être réglé à 16 ms

# Pont VSWR

J'utilise actuellement un pont Stockton avec une ferrite à noyau binoculaire. Cela ne nécessite aucun réglage de "balance" et donne une mesure de puissance étalonnée .

J’ai calculé (voir note) que la tension de ligne RMS = 0.0837N où N = lecture ADC Arduino.

(D'après la feuille de calcul du pont de Stockton : Une entrée de 3,3 V correspond à 147 W, soit 85,7 Vrms. Lecture ADC N = 1024\*Vin/3.3 Donc Vrms=85.7N/1024 = 0.0837N)

Puissance = Vrms2/50; VSWR = (Vf+Vr)/(Vf-Vr)

# Stored Tune Solution Data Structures

Approche proposée : 3 blocs de mémoire EEPROM - un pour chaque antenne.

Les solutions sont stockées tous les 10KHz, à partir de 0 ; chaque antenne stocke les solutions pour toutes les fréquences de 0 à 61,49MHz (juste au-dessus du taux de Nyquist pour les radios HPSDR). Les solutions sont numérotées de 0 à 6149 ; le nombre total est de 6150.

Il y a 3 octets par solution stockée :

|  |  |
| --- | --- |
| **byte** | **Meaning** |
| 0 | bit0=1: no data; bit0=0: data OK  bit7=1: high Z; bit7=0: low Z |
| 1 | Inductance word |
| 2 | capacitance word |

Adresse de la solution dans l'EEPROM, avec la 1ère antenne = antenne 1 :

(Antenne-1)\*32768 + 3\*Int(Freq\_in\_KHz/10KHz)

Ainsi, pour l'antenne 2 à 61,471 MHz, la solution sera à l'adresse décimale 57356.

Nous avons besoin d'une EEPROM stockant au moins 128Koctets.

Pour la version autonome avec affichage Nextion, 3 variables supplémentaires sont stockées :

|  |  |
| --- | --- |
| **address** | **meaning** |
| 0x1FFF0 | Page d'affichage en cours d'utilisation. Valeurs légales 1-4. Si autre valeur, utiliser la page 4 |
| 0x1FFF1 | Affichage de la crête/moyenne. Bool. |
| 0x1FFF2 | ATU Activé/désactivé. Bool. |
| 0x1FFF3 | Échelle d'affichage. 0=100W ; 1=200W ; 2=500W ; 3=1000W ; 4=2000W |
| 0x1FFF4 | Autoriser l'accord rapide. !=0 pour autoriser l'accord rapide. |

## Données locales / structures de données

Approche proposée : stocker les solutions pour l'ensemble de la bande HF dans la SRAM pour chaque antenne.

1. Si l'antenne change, il faut charger un bloc différent à partir de l'EEPROM.  
   a. Stockage requis ~18Kbyte. Impraticable avec un AVR 8 bits, il faut encore un Nano 33.  
   b. Lire un nouveau lot chaque fois que l'antenne change  
   c. Temps de lecture ~165ms ; ce qui signifie qu'il y a une période significative pendant laquelle le TX n'est pas possible.

|  |  |  |
| --- | --- | --- |
| unsigned int | GTunedFrequency10 | Frequency the ATU is tuned to, in units of 10KHz |
| bool | GATUEnabled |  |
| unsigned int | GQueuedCATFrequency | 0xFFFF: no frequency stored  0-6149: ATU frequency passed by Thetis during TX |
| byte | GAntenna | Antenna number (1-3) =0 if not set |
| bool | GFrequencySet | True if a frequency has been set by THETIS |
| bool | GPCTuneActive | If true, a TUNE is happening |
| bool | GPTTPressed | If true, PTT is pressed |
| bool | GTXAllowed | True if ATU solution should be sent when TX asserted |
| bool | GATUIsTuned | True if valid solution already set |
| byte\* | GSolutionBuffer | Block of tuning solutions buffered in internal memory  This holds full data for one antenna |



Figure 6: EEPROM Memory map

# Version externe d’Aries

Tout comme intégré à un transceiver Andromeda, on peut utiliser Aries comme une ATU externe, en ajout pour les radios existantes. Nous devrions modifier Thetis pour acheminer la RF toujours par ANT1, et avoir un commutateur T/R externe et 2 relais de sélection ANT.

4 bits supplémentaires sont envoyés au registre à décalage SPI pour contrôler ces relais..

# Software Algorithms

## Event Response

|  |  |
| --- | --- |
| **Event** | **Action Required** |
| Lorsque TX est activé | Si la solution existante est déjà pilotée : pas d'action  Si la nouvelle solution est prête à être mise en œuvre : envoyer la solution aux relais.  Si le drapeau " Tune in progress " est activé : commencer l'accord, après le délai "TT on".  Mettre l'indicateur " TX in progress ". |
| Lorsque TX est desactivé | Effacer le drapeau “TX in progress”  Effacer le drapeau “tune in progress”  Si l'algorithme d'accord est en cours, mettre fin à l'algorithme |
| Lorsque la nouvelle fréquence est reçue | S'il est déjà en TX - attendre la fin de TX  Sélectionner la solution la plus proche ou bypass  Envoyer un message CAT avec la solution disponible/non disponible |
| Lorsqu'une nouvelle antenne TX est reçue | Temporarily set bypass “null solution”  Lire un nouveau bloc de données dans l'EEPROM  Vérifier les solutions d'accord pour l'antenne N autour de la fréquence actuelle  Sélectionner la solution la plus proche ou Bypass  Envoi d'un message CAT avec la solution disponible/non disponible |
| Lorsque la validation ATU est reçue | Traiter comme une nouvelle antenne |
| Lorsque ATU Disable est reçu | Régler l’ATU sur “bypass”  Définir le drapeau |
| Lorsque le message "TUNE start" est reçu | Mise en place du drapeau " tune in progress ".  Si TX est déjà affirmé, commencer l'algorithme |
| Lorsque le message "TUNE end" est reçu | Ignorer - cela est signalé par le retrait du PTT |
| Lorsque l'algorithme TUNE est termin | Stocker la solution dans l'EEPROM  Renvoyer “success” après stockage |
| Si le message L/C fine tune CAT a été reçu | Ajuster le réglage L/C  Ne pas ré-enregistrer |
| Lorsque l'effacement est reçu pour l'ant N | Effacer le bloc de données pour cette antenne  Si la même antenne est sélectionnée :  définir la solution à utiliser = bypass  effacer la copie RAM des solutions de réglage  Envoyer le message de réponse |

## Original Search Algorithm

Do the following while GTuneActive is true:

Quick tune:

1. Try “fine step L and C” around current setting
2. If that fails select full tune

Full Tune:

1. On the basis of frequency, select min/max L and C and coarse, mid step size
2. Select “Low Z”
3. Step through C values at coarse step, find step with min VSWR & achieved VSWR
4. Select “High Z”
5. Step through L values at coarse step, find step with min VSWR & achieved VSWR
6. If best result is low Z:
   1. Re-select Low Z
   2. Select best C value
   3. Step through L values at coarse step, find step with min VSWR & achieved VSWR
   4. Select best L value
   5. Step through C values +/- 2 coarse steps at mid step for min VSWR
   6. Select best C value
   7. Step through L values +/- 2 coarse steps at mid step for min VSWR
   8. Select best L value
   9. Step through C values +/- 2 mid steps at step=1 for min VSWR
   10. Select best C value
   11. Step through L values +/- 2 mid steps at step=1 for min VSWR
   12. Select best L value
7. Else if best result was high Z
   1. Select best L value
   2. Step through C values at coarse step, find step with min VSWR & achieved VSWR
   3. Select best C value
   4. Step through L values +/- 2 coarse steps at mid step for min VSWR
   5. Select best L value
   6. Step through C values +/- 2 coarse steps at mid step for min VSWR
   7. Select best C value
   8. Step through L values +/- 2 mid steps at step=1 for min VSWR
   9. Select best L value
   10. Step through C values +/- 2 mid steps at step=1 for min VSWR
   11. Select best C value
8. Store result to EEPROM if final VSWR < 1.5
9. Report success/fail and end



At each tick:

1. Check if “terminate” signal
   1. Exit if so with no result saved
2. Measure VSWR
3. If VSWR < min already achieved
   1. Record step and VSWR
4. If next step > end:
   1. move to next state
5. Else
   1. set next step
   2. drive solution

To find next candidate solution:

If (setting == end)

Signal end;

Else

New setting = constrain (setting+step, min, max);

The current algorithm executes potentially 144 search steps:

* Initial Quick tune:
  + 16 fine steps L
  + 16 fine steps C
* Full tune, if quick fails:
  + 16 coarse C, low impedance
  + 16 coarse L, high impedance
  + 16 coarse L or C depending on best match
  + 16 mid step L
  + 16 mid step C
  + 16 fine step L
  + 16 fine step C
* Note that:
  + For low frequencies we can accept bigger steps
  + For high frequencies we can limit the search space

## New Search Algorithm

With real antennas the algorithm does not always find a match, or the best match. The “narrow down” process seems OK but the initial search isn’t comprehensive.

|  |  |
| --- | --- |
| Current |  |
| Required |  |
| And Potentially |  |
| And for some loads |  |

Suggested approach:

* Keep the 2 stage algorithm
* Have each part table driven, so the search range can be set by frequency
* For the first part, have the number of search steps set by the table
* Potentially that might allow user settable search algorithms to trade speed & performance
* Stage 1a: perform sweeps of one parameter according to table
* Stage 1b: for the minimum found, perform a sweep of the opposite parameter
* Stage 2a: medium sweep 1st parameter
* Stage 2b: medium sweep 2nd parameter
* Stage 2c: fine sweep 1st parameter
* Stage 2b: fine sweep 2nd parameter
* In all cases you swap over the parameter being swept at each stage
  + 1st table:
    - Upper frequency
    - Stage 1a Start Row for this frequency
    - Number of rows for this frequency
    - Max L, Max C
    - Stage 1b min, max, step
    - Stage 2 mid step +/- range, step size
    - Stage 2 fine step +/- range
    - (the entire algorithm can be changed by replacing this table with no code change)
  + 2nd table with Parameters for stage 1a:
    - High/Low Z
    - Sweeping L/C
    - Fixed value
    - Min, max, step value
* To be recorded in 1st stage, to find min
  + C value
  + L value
  + High/Low Z value
  + VSWR value
  + (it doesn’t matter which sweep we find it on)



# Testing

## Test Loads

|  |  |  |  |
| --- | --- | --- | --- |
| Switch position | VSWR | Resistance | Construction |
| 1 | 8:1 Low | 6.25R | 16x100R parallel |
| 2 | 5:1 Low | 10R | 10x100R parallel |
| 3 | 4:1 Low | 12.5R | 8x100R parallel |
| 4 | 3:1 Low | 16.6R | 6x100R parallel |
| 5 | 2:1 Low | 25R | 4x100R parallel |
| 6 | 1:1 | 50R | 4x100R parallel, in series with 4 more 100R parallel |
| 7 | 2:1 High | 100R | 2x100R parallel, in series with 2 more 100R parallel |
| 8 | 3:1 High | 150R | 3x100R series, parallel with another 3x100R series |
| 9 | 4:1 High | 200R | 2x100R series  (4x100R in series, parallel with 4x100R series) |
| 10 | 5:1 High | 250R | 2x100R series, +2x100R parallel |
| 11 | 8:1 High | 400R | 4x100R series |

These need to be good for around 5W

# Protection de l’Amplificateur

Outre le fait d’offrir un affichage autonome, il a été souhaité d’ajouter une fonction simple de protection de l'amplificateur. Cela permettra à Aries de faire partie d'un ensemble PA évolué, connecté à Thetis mais dont les commandes d'antenne ne sont pas gérées par Thetis. Il détectera les conditions de défaut (VSWR élevé, puissance inverse élevée...) et coupera le PA presque instantanément en utilisant un simple flip flop ; il permettra à l'Arduino de surveiller les conditions de défaut et de montrer un affichage "déclenché" quand ils se sont produits. Le connecteur J1 connecte Aries à la carte de protection.  
  
La carte de protection possède des comparateurs pour ses entrées analogiques. Si l'une d'entre elles dépasse le seuil, la sortie du comparateur passe à l'état bas. Les 4 sorties sont reliées par un ANDED ; toute sortie supérieure au seuil fait passer la porte AND à l'état bas et déclenche une bascule S/R. La sortie de la bascule est à l'état bas, ce qui supprime le PTT de l'amplificateur. La sortie de la bascule qui est basse supprime l'amplificateur PTT et sera surveillée par l'Arduino (DIG8). La bascule peut être réinitialisée en activant le signal Reset haut puis bas (également nécessaire après la mise sous tension).  
  
U1.A détecte un ROS "indicatif" (il n'est pas précis) ;

- U1.B détecte un excès de puissance inverse ;

- U1.C détecte l'excès de puissance d’excitation ;

- U1.D (optionnel) détecte l'excès de température.

Les signaux numériques utilisés pour le circuit de protection proviennent d'un MCP23017 sur le bus I2C. DIG8 permet également de surveiller l'état de la bascule. Lorsqu'un déclenchement se produit, une nouvelle page d'affichage Nextion doit être affichée. Cette page interrogera périodiquement les 4 entrées de condition de défaut et, lorsqu'elle est OK, activera un bouton de "réinitialisation". Lorsque l'on appuie sur le bouton de réinitialisation, la réinitialisation de la bascule doit être cyclique et le fonctionnement normal doit reprendre.

Les signaux du MCP23017 sont les suivants :

|  |  |  |
| --- | --- | --- |
| **Pin** | **State** | **Description** |
| GPB0 | Input | VSWR comparator input. 0 if tripped |
| GPB1 | Input | Reverse power comparator input. 0 if tripped |
| GPB2 | Input | Drive power comparator input. 0 if tripped |
| GPB3 | Input | PA Temperature comparator input. 0 if tripped |
| PB4 | Input | unused |
| GPB5 | Input | unused |
| GPB6 | Input | unused |
| GPB7 | Input | Low if 6M band selected. Display “6M” |
| GPA0 | Output | Protection Flip Flop Reset. =0: normal state; =1: applies reset |
| GPA1 | Output | PA PTT enable. =1: normal operation. =0: PTT disallowed. |
| GPA2 | Input | Low if 10M band selected. Display “10M” |
| GPA3 | Input | Low if 15M band selected. Display “17+15M” |
| GPA4 | Input | Low if 20M band selected. Display “30+20M” |
| GPA5 | Input | Low if 40M band selected. Display “60+40M” |
| GPA6 | Input | Low if 80M band selected. Display “80M” |
| GPA7 | Input | Low if 160M band selected. Display “160M” |

MCP23017 base address = 0x20. MCP23017 registers are accessed in the mode where IOCON.BANK=0:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Register** | **Address** | **Purpose** | **Required State** | **Initial Value** |
| IODIRA | 0x00 | I/O direction. Per bit; 1 = input | 0,1=o/p; 2-7 = i/p | 0xFC |
| IODIRB | 0x01 | I/O direction. Per bit; 1 = input | All inputs | 0xFF |
| IOPOLA | 0x02 | I/O polarity. 0 = not inverted | All not inverted | 0x00 |
| IOPOLB | 0x03 | I/O polarity. 0 = not inverted | All not inverted | 0x00 |
| GPINTENA | 0x04 | Interrupt on change. 1 = enabled | No interrupts | 0x00 |
| GPINTENB | 0x05 | Interrupt on change. 1 = enabled | No interrupts | 0x00 |
| DEFVALA | 0x06 | Interrupt compare. Per bit | Don’t care | 0x00 |
| DEFVALB | 0x07 | Interrupt compare. Per bit | Don’t care | 0x00 |
| INTCONA | 0x08 | Interrupt on change control. Per bit. | Don’t care | 0x00 |
| INTCONB | 0x09 | Interrupt on change control. Per bit. | Don’t care | 0x00 |
| IOCON | 0x0A | I/O configuration | Bank=0; mirror=0;  Seqop=0; DISSLW=1; ODR=1; INTPOL=0; | 0x14 |
| IOCON | 0x0B | Same register |  | Ignore |
| GPPUA | 0x0C | Pullup enable. 1 = enabled; per bit | All 1 | 0xFF |
| GPPUB | 0x0D | Pullup enable. 1 = enabled; per bit | All 1 | 0xFF |
| INTFA | 0x0E | Interrupt flag; readonly |  | n/a |
| INTFB | 0x0F | Interrupt flag; readonly |  | n/a |
| INTCAPA | 0x10 | Interrupt capture; readonly |  | n/a |
| INTCAPB | Ox11 | Interrupt capture; readonly |  | n/a |
| GPIOA | 0x12 | GPIO read register | b1=0 (disables PTT) | R/W |
| GPIOB | 0x13 | GPIO read register |  | R/W |
| OLATA | 0x14 | GPIO write register |  | 0x00 |
| OLATB | 0x15 | GPIO write register |  | 0x00 |

When amplifier protection is provided, there are two analogue inputs (A6 and A7). A6 measures PA current; A7 is not used.

Kjell’s figures: Input = 0 => current = 0A; Input = 3.12V => current = 20A

Therefore full scale ADC reading = 21.154A. Current = 21.154N/4096 = 0.00051645N

Potential conflicts with use of I2C bus:

* Erase – essentially the process just waits until the erase complete, ignoring normal processing. So no conflict.
* Read block – this happens in a single event; the processor may miss a few ticks. No conflict.
* Write solution – this happens in a single event. No conflict.

The protection code is enabled automatically. The MCP23017 is detected during initialise, and if detected protection mode is automatically selected.

# Display

L'approche originale utilisait un écran LCD, interfacé I2C. Mais l'ajout du mode autonome nécessite des écrans plus "appropriés" et un afficheur Nextion de 3,2" est maintenant utilisé.  
Un problème intéressant a été trouvé : la tentative de définir la couleur du texte affiché en utilisant Set\_font\_color\_pco() a entraîné le plantage de l'affichage ou du code de la bibliothèque d'affichage.   
Tous les écrans ont un bouton activé/désactivé et une chaîne d'état ATU. Cela changera entre "disabled", "no tune", Tuned" et "tuning". Certains affichages ont un bouton de puissance moyenne/de pointe.

|  |  |
| --- | --- |
| Affichage de la puissance en mode “bargraph” |  |
| Affichage analogique de la puissance |  |
| Affichage à aiguilles croisées  (affecté par un certain “flicker”) |  |
| Écran “ingénierie”  Cet affichage est destiné au débogage. Vous pouvez voir tous les différents "chiffres" du code. Vous pouvez également affiner le réglage de l'ATU. |  |
| Cette page s'affiche si le circuit de protection détecte une condition de déclenchement.  Les 4 zones de texte sur la droite affichent l'état de déclenchement ou de non-déclenchement.  Lorsque les 4 zones ne sont pas déclenchées, le bouton de test (en bas à droite) devient “Reset” et lorsqu'il est actionné, le fonctionnement normal reprend. |  |

# Arduino Libraries

## Board support

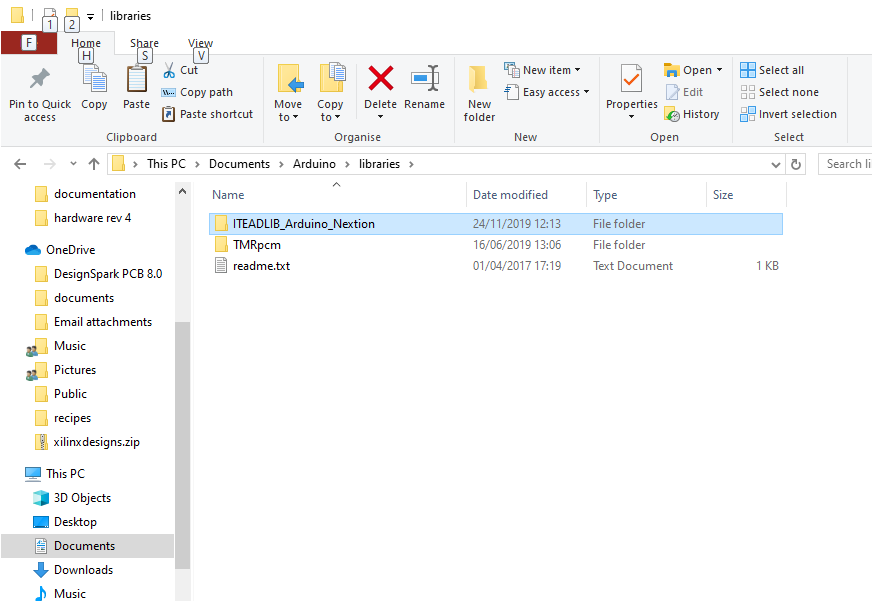
Install “Arduino SAMD boards (32 bit ARM Cortex-M0+) by Arduino”



## Nextion Library

There are issues using this library with an Arduino nano 33 IoT: any of the “read back” calls fail and hang the Arduino. There is an alternative library available that I’ve not yet tried that could fix that if required (https://github.com/jyberg/Enhanced-Nextion-Library)

The library needs to be installed using a similar process:

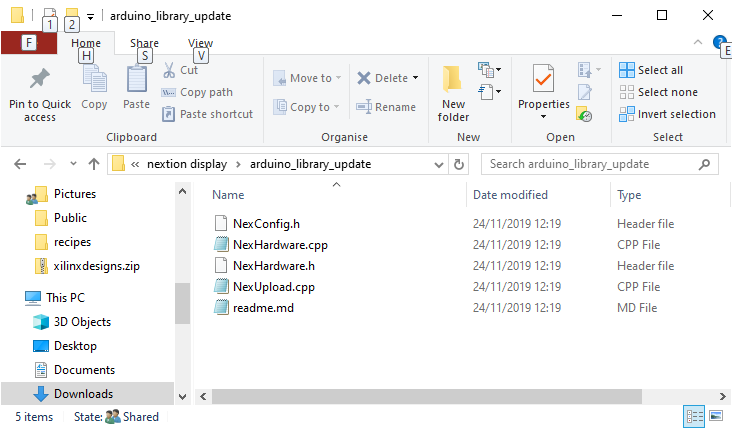
1. Visit the repository on github: <https://github.com/itead/ITEADLIB_Arduino_Nextion>
2. Click “clone or download” then “download zip”
3. Store the zip file on your PC for example in the “downloads” folder
4. Open the zip file and extract all files. You will now have a folder “ITEADLIB\_Arduino\_Nextion-master” which will hold one folder also called “ITEADLIB\_Arduino\_Nextion-master”
5. Rename the second folder “ITEADLIB\_Arduino\_Nextion” (remove the “-master” part)
6. Copy that whole folder to your “documents\arduino\libraries” folder
7. (This is the library published by the display manufacturer. Be aware there is some foul language in the "html" folder - delete the entire "html" folder if you do not want that)
8. Your “documents\arduino\libraries” folder should now have that library:
9. 

## Patch the ITEADLIB Library

Quatre fichiers (plus un fichier readme) doivent être copiés depuis le dépôt Aries vers le dossier ITEADLIB dans les bibliothèques Arduino.

1. Ouvrez le dossier "nextion display\arduino\_library\_update"

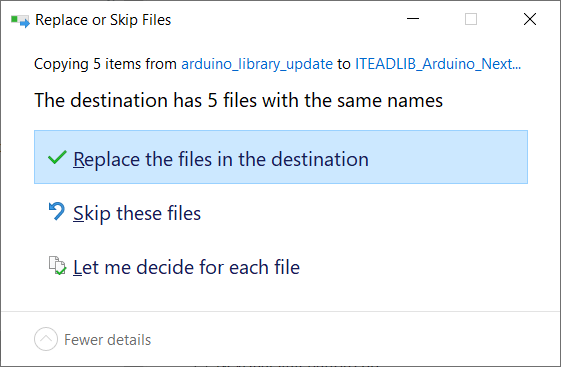
2. Il contiendra les fichiers suivants :



1. Sélectionner puis copier ces fichiers

2. Naviguez vers votre dossier "documents\arduino\libraries\ITEADLIB\_Arduino\_Nextion"

3. Collez les 5 fichiers dans ce dossier. Assurez-vous de remplacer les fichiers originaux.

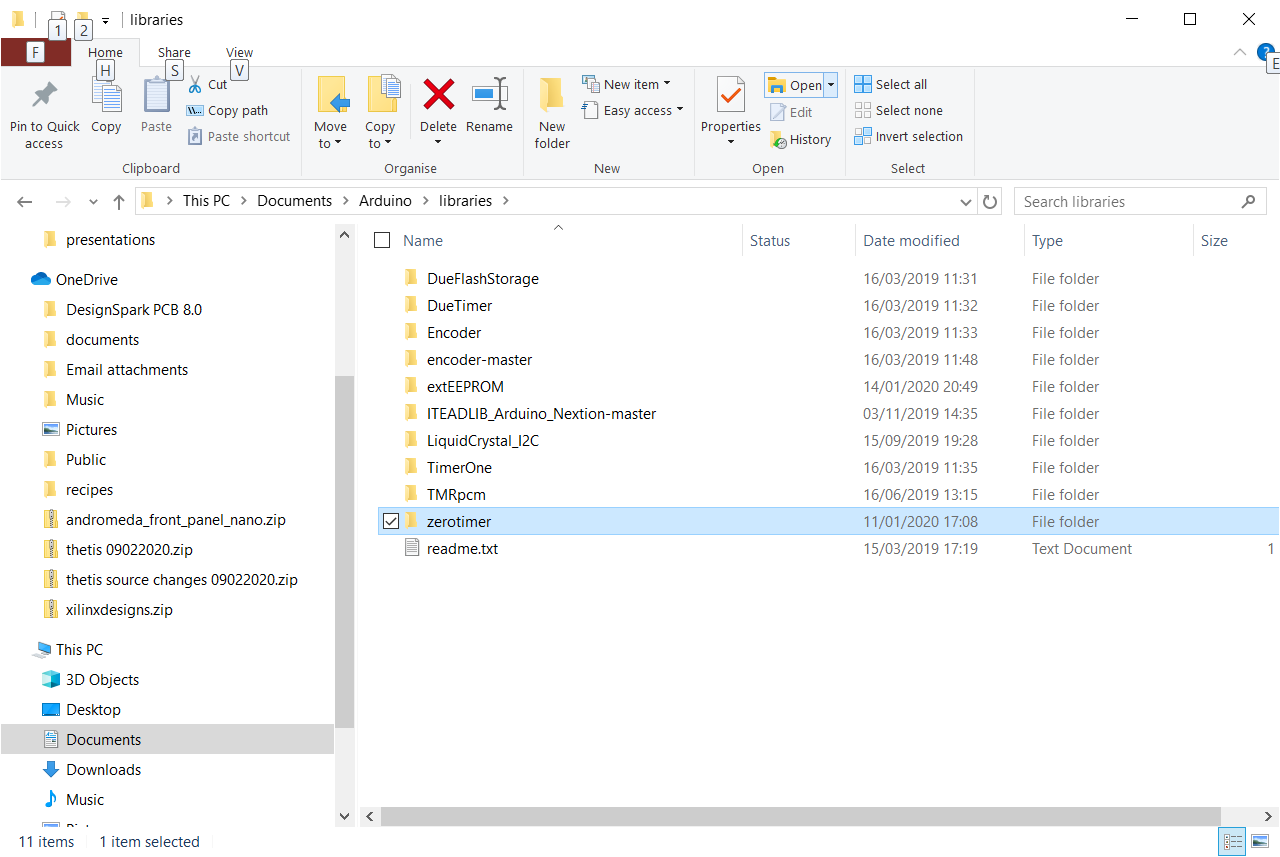


1. 4 existing files will be replaced and the readme file will be added.

## ZeroTimer Library

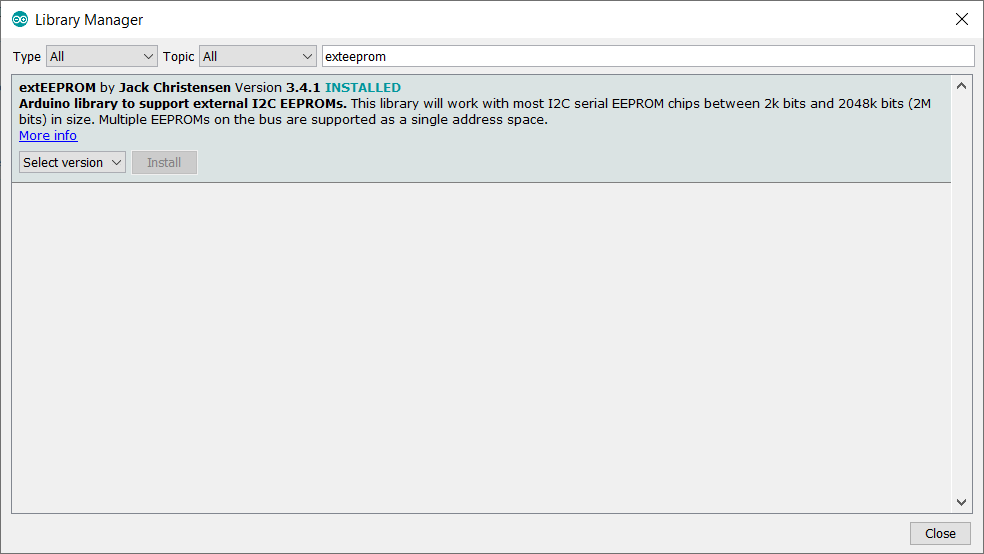
1. Download from <https://github.com/EHbtj/ZeroTimer>
2. Open the zip file and extract all files. You will now have a folder “ZeroTimer-master” which will hold one folder also called “ZeroTimer-master”
3. Rename the second folder “ZeroTimer” (remove the “-master” part)
4. Copy that whole folder to your “documents\arduino\libraries” folder

Your Arduino libraries folder will now include ZeroTimer:



## extEEPROM Library

1. click Tools > Manage Libraries…
2. The “installed libraries” form opens
3. Type “exteeprom” into the bar at the top
4. The library “extEEPROM” should be shown. Click Install
5. The library should show “installed”



## Compiling

Select Tools > Board > Arduino NANO 33 IoT

Click the “tick” icon to compile

Click the “right arrow” icon to download

There are some “conditional compile” options enabled using #define options in globalinclude.h

|  |  |
| --- | --- |
| #define SWVERSION nn  #define HWVERSION 5  #define PRODUCTID 2 | These are standard settings that can be displayed in Thetis. PRODUCTID must be set to 2.  It is important that rev 5 boards have HWVERSION set to 5. |
| #define VSWR\_SWAPVFVR | If defined, the Vf and Vr functions are swapped. This depends on the type of VSWR bridge. With rev 5 PCBs and binocular core, this should be commented out (//#define VSWR\_SWAPVFVR) |
| #define CONDITIONAL\_ALG\_DEBUG | If defined, additional debug messages are displayed to the serial console. This cannot be defined when connected to thetis! |
| #define CONDITIONAL\_ALG\_SIMVSWR | If defined, VSWR readings are simulated. This allows algorithm tests with no RF present. Must NOT be defined for normal use. |
| #define CONDITIONAL\_STREAM\_ADCREADINGS | If defined, Vf and Vr readings are streamed to the serial console. This is used for VSWR bridge debugging and must NOT be enabled in normal use. |
| #define VDISPLAYSCALE n  #define VNUMPAGES n | One less than the number of display scales available (when set to 4, 5 scales can be used)  The number of “normal” display pages available |

# To Do

Modification du code à ajouter :

* Sélection de la vitesse I2C en fonction de l'autonomie ou non (400KHz si autonome, avec MCP23017 supplémentaire) (pour l'instant, la vitesse était déjà fixée à 400KHz).

# Modifications du PCB Rev 5

RL18 n'a pas la bonne empreinte de PCB. Des changements de câblage sont nécessaires :A screenshot of a computer

Description automatically generated with medium confidence

Il y a un court-circuit à la masse sur le signal "REV" sur le côté supérieur de la carte entre R13 et L20 près de J12 et F1. Ceci est dû au fait que la piste est trop proche d'un via de masse. La piste doit être coupée légèrement pour enlever la connexion au via. (ndt : ce problème est corrigé avec la version “rev5\_28\_01\_21” du pcb)

