INSTITUTO UNIVERSITARIO AERONÁUTICO
Departamento Mecánica Aeronáutica

REVISIÓN MICROCONTROLADORES PARA COMPUTADORA DE MÓDULO DE COMANDO PARA PARACAÍDAS

Informe técnico: DMA-001/18 **Revisión:** /

Proyecto: /

Fecha: 22/03/18

Autor:

Ing. Diego Llorens
Investigador

Revisó:

Nombre del revisor
Cargo del revisor

REVISIÓN MICROCONTROLADORES PARA COMPUTADORA DE MÓDULO DE COMANDO PARA PARACAÍDAS

Por:

Ing. Diego Llorens

RESUMEN

En el marco del proyecto PIDDEF 038/14 – “Paracaídas Comandado Autónomo”, se realizó una revisión de micro controladores y sus respectivas placas de desarrollo para proponer actualizar la computadora de vuelo del módulo de comando para paracaídas guiados.

Se presenta una comparación de 4 micro controladores en la que se analiza la factibildad de cada uno de ellos para reemplazar el hardware actual de autopiloto y la posibilidad de ampliación futura del sistema mediante nuevos periféricos. Además se realiza un análisis del esfuerzo para migrar el programa de autopiloto existente al nuevo hardware.

Se determinó que la placa de desarrollo Teensy 3.6 para un micro controlador de arquitectura Cortex-M4F (NXP K66) es una opción viable para mejorar el hardware disponible con un esfuerzo bajo de modificación del código fuente disponible.

Córdoba, 22 de marzo de 2018

ÍNDICE

SISTEMA DE REFERENCIA Y CONVENCIÓN DE SÍMBOLOS

Símbolo	Unidad	Descripción
ADC	-	Analog to Digital Converter
CAN	-	Controller Area Network
DAC	-	Digital to Analog Converter
FPU	-	Floating Point Unit
GPIO	-	General Purpose Input Output
IMU	-	Inertial Measurement Unit
I^2C	-	Synchronous Multi-master Multi-slave Packet switched Single-ended Serial Computer Bus
MCU	-	Microcontroller Unit
PWM	-	Pulse Width Modulation
SO	-	Sistema Operativo
SPI	-	Serial Peripheral Interface
SRAM	-	Static Random Access Memory
UART	-	Universal Asynchronous Receiver-Transmitter
USART	-	Universal Synchronous and Asynchronous Receiver-Transmitter
TTL	-	Transistor–Transistor Logic

1. INTRODUCCIÓN

El desarrollo del autopiloto para paracaídas comandado está basado en el hardware abierto ArdupilotMega 1.0 [1] y el paquete de librerías desarrollado para dicho hardware (ArdupilotMega v1.02). Sobre esta plataforma se realizó el desarrollo de un código de autopiloto propio para el control, guiado y navegación de paracaídas comandados [2]. Si bien las librerías y el software generado para este hardware puede seguir siendo ampliado y mejorado, existe, actualmente, la limitación que las placas de autopiloto no se producen más; el desarrollo de las mismas ha evolucionado a versiones más compactas y con arquitectura diferente tanto a nivel del micro controlador como de los sensores integrados [3][4].

Por una parte se busca disponer de un micro controlador más potente que tenga capacidad de ampliación de las funcionalidades de operación del módulo de comando y que además permita utilizarlo como plataforma de desarrollo para otras variantes de autopiloto que puedan surgir en el futuro. Y por otro lado se requiere que la placa de desarrollo en donde este integrado el micro controlador sea suficientemente genérica para tener flexibilidad a la hora de tener que integrar nuevos sensores o incorporar nuevas funcionalidades a nivel de hardware, eliminando así la posibilidad de depender de una sistema de hardware integrado en particular.

Teniendo en cuenta estos lineamientos se plantean a continuación los requerimientos tanto a nivel de hardware como a nivel de software (librerías) que debe tener una posible placa de desarrollo que sustituya la hardware actual y que sirva para realizar el desarrollo de la computadora de vuelo para el módulo de comando para paracaídas comandados.

2. DESARROLLO

2.1 Requerimientos de software

La implementación del software de piloto automático para paracaídas comandados utiliza librerías tanto de Arduino [5] como de AVR [6] para implementar funcionalidades y la interacción con el hardware del microcontrolador y los sensores. La dependencia del programa de auotpiloto con estas librerías se puede visualizar mediante un esquema de capas en donde el nivel de dependencia se da de arriba hacia abajo (las capas superiores requieren de las capas inferiores para funcionar) estando el hardware físico en la base del esquema (ver Figura 1).

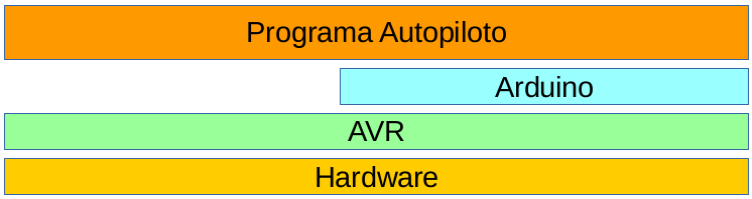

Figura 1: Capas de software del programa de autopiloto

Para poder hacer una migración progresiva del programa de autopiloto a una nueva versión de hardware es deseable que este disponga de una paquete de librerías compatible con Arduino para tener que realizar, solamente, pequeños cambios en la porción de código que depende de las librerías de AVR; o que el hardware sea compatible con el conjunto de librerías de AVR, en cuyo caso sería compatible con las librerías de Arduino.

2.2 Requerimientos de hardware

A nivel de hardware se requiere que se puedan mantener las funcionalidades actuales de la placa de autopiloto y que exista alguna posibilidad de ampliación futura por medio de mayor cantidad de interfaces de comunicación y puertos analógicos y digitales. Para ello, se listan a continuación los servicios, sensores y periféricos (ver Tabla 1 y Tabla 2) que actualmente están disponibles en la placa de autopiloto y cuáles son necesarios para el control del módulo de comando de paracaídas. Además, se listan la cantidad de pines utilizados para todos estos servicios. Con estos listados se determina que el hardware tiene que tener las siguientes interfaces de comunicación y pines analógicos y digitales:

- UART (al menos 3)
- SPI
- I^2C
- 7 puertos analógicos para entrada de señales PWM
- 15 puertos digitales para salida de señales PWM

Sensor	Interfaz / comunicación	Requerido para paracaídas	Pines del MCU
Acelerómetro (para IMU)	Por medio de ADC externo 7844	NO	N/a
Girómetro (para IMU)	Por medio de ADC externo 7844	NO	N/a
Barómetro	I^2C	SI	2 (SDA + SCL)
GPS	UART	SI	2 (Serial2: Tx + Rx)
Magnetómetro	I^2C	SI	2 (SDA + SCL)
Sensor de distancia laser	I^2C	SI	2 (SDA + SCL)
Tensión y corriente de batería	Por medio de ADC interno MCU	SI	2 (A0 y A1)

Tabla 1: Lista de sensores del hardware de autopiloto

Dispositivo / servicio	Interfaz / comunicación	Requerido para paracaídas	Pines del MCU
ADC 7844	SPI	NO	4 (MISO + MOSI + SCK + CS)
Telemetría	UART	SI	2 (Serial1: Tx + Rx)
Memoría de logueo (flash)	SPI	SI	4 (MISO + MOSI + SCK + CS)
Consola	UART	SI	2 (Serial3: Tx + Rx)
Sistema de control	Entradas PWM	SI	3 (actuadores + motor) 1 (control manual-automático) 1 (selección modo vuelo)
Sistema de control	Salidas PWM	SI	3 (actuadores + motor)

Tabla 2: Lista de servicios del programa de autopiloto

2.3 Listado de placas de desarrollo

Se analizarán tres propuestas de placas de desarrollo con diferentes micro controladores para determinar una opción para reemplazar la computadora actual que se dispone para el módulo de comando para paracaídas autónomo.

2.3.1 Placas de desarrollo de Arduino.cc

La primera opción está orientada a utilizar una placa de desarrollo con micro controlador de la familia Arduino para tener 100% de compatibilidad con las librerías de Arduino. Se presentan dos opciones a continuación: ArduinoMega (Figura 2) que tiene integrado el mismo microcontrolador que se está utilizando en el hardware actual del autopiloto (ATMega 2560) y Arduino M0 (Figura 3) que es una versión actualizada del Arduino UNO con micro controlador ARM Cortex M0+ con mayor capacidad de cálculo. En la Tabla 3 se presenta un resumen de las especificaciones técnicas, las interfaces de comunicación y la cantidad de pines analógicos y digitales que disponen estas placas.

Figura 2: Placa de desarrollo Arduino Mega

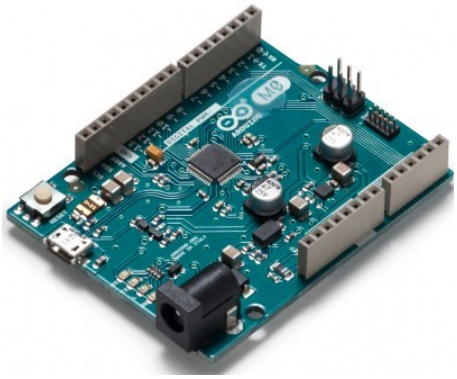

Figura 3: Placa de desarrollo Arduino M0

Placa de desarrollo	Arduino Mega	Arduino M0
MCU	ATMega 2560	SAMD21 ARM Cortex M0+
Arquitectura	8 bit	32 bit
Velocidad de reloj	16 MHz	48 MHz
Memoria Flash	256 kB	256 kB
SRAM	8 kB	32 kB
FPU	ninguno	ninguno
SO	ninguno	ninguno
Voltaje operación	5 V	3,3 V
GPIO	54	20
Salida PWM	15	12
Entrada analógica / ADC	16 / 10 bit	6 / 12bit
Salida analógica / DAC	ninguno	1 / 10 bit
UART	ninguno	1
USART	4	3
I²C	1	1
SPI	Si	Si
CAN	No	No

Tabla 3: Especificaciones técnicas, de comunicación y I/O de las placas de desarrollo elegidas de Arduino.cc

2.3.2 Placa de desarrollo de PJRC

Esta placa de desarrollo está montada en torno a un micro controlador ARM Cortex-M4 con unidad de punto flotante. La unidad de punto flotante tiene precisión de 32 bit para operaciones de simple precisión; con esta unidad las operaciones matemáticas se realizan unas 30 veces más rápidas que utilizando una librería que emule las operaciones con este tipo de datos (por software). Posee, además, un lector para tarjetas de memoria SD integrado a la placa y es compatible con las librerías de Arduino. En la Tabla 4, a continuación, se presentan las especificaciones técnicas de la placa de desarrollo.

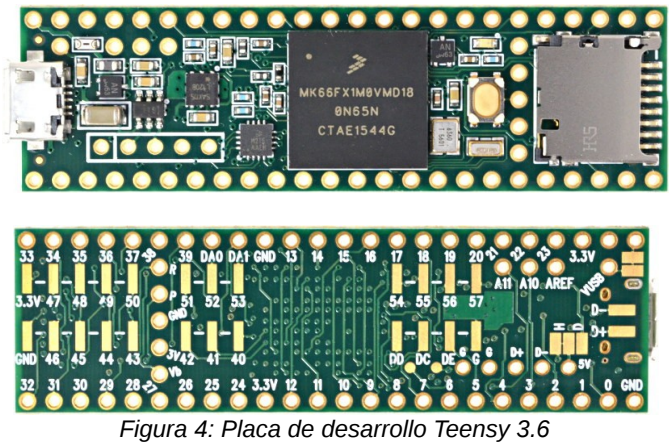

Figura 4: Placa de desarrollo Teensy 3.6

Placa de desarrollo	Teensy 3.6
MCU	NXP MK66FX1M0VMD18 Cortex-M4F
Arquitectura	32 bit
Velocidad de reloj	180 MHz
Memoria Flash	1024 kB
SRAM	256 kB
FPU	si
SO	ninguno
Voltaje operación	3,3 V
GPIO	58
Salida PWM	6 / 16 bit
Entrada analógica / ADC	25 / 13 bit
Salida analógica / DAC	2 / 12 bit
UART	ninguno
USART	6
I^2C	4
SPI	Si
CAN	2

Tabla 4: Especificaciones técnicas, de comunicación y I/O de las placa de desarrollo Teensy 3.6

2.3.3 Placa de desarrollo Pycom

Estas placas de desarrollo están orientadas al desarrollo de sistemas autónomos interconectados mediante algún protocolo de comunicación. Entre las opciones de comunicación disponibles para estas placas se encuentran: Wifi, LoRa, Bluetooth y Sigfox. Además, hay disponibles algunos módulos de expansión con diferentes sensores integrados para facilitar el desarrollo de prototipos. Las placas de desarrollo incorporan un microcontrolador de la empresa Espressif ESP32 de dos núcleos. A continuación, en la Tabla 5, se listan las especificaciones técnicas del módulo.

Figura 5: Placa de desarrollo LoPy 4

Placa de desarrollo	LoPy 4
MCU	Xtensa LX6
Arquitectura	32 bit x 2
Velocidad de reloj	160 MHz
Memoria Flash	8192 kB
SRAM	520 kB
FPU	si
SO	ninguno
Voltaje operación	3,3 V
GPIO	24
Salida PWM	18 / s/d bit
Entrada analógica / ADC	18 / s/d bit
Salida analógica / DAC	2 / s/d bit
UART	3
USART	ninguno
I^2C	2
SPI	Si
CAN	1

Tabla 5: Especificaciones técnicas, de comunicación y I/O de las placa de desarrollo LoPy 4

2.4 Análisis de las placas seleccionadas

2.4.1 Entradas y salidas digitales y analógicas

La cantidad de entradas analógicas requeridas en el hardware es de 7. Dos de ellas se utilizan para leer el estado (tensión) de una batería de alimentación y la otra para registrar el consumo de corriente. Las cinco restantes se utilizan para para leer las señales PWM provenientes de los canales del equipo de control R/C. Todas las placas, excepto el modelo Arduino M0 (que dispone de 6 entradas analógicas), satisfacen este requerimiento. En el prototipo final del módulo de comando, probablemente no sea necesario disponer de entradas analógicas conectadas a un receptor de radio para realizar el control remoto del paracaídas y, por lo tanto, no sea una limitación. Sin embargo, si se implementa un prototipo intermedio que disponga de esta capacidad de control remoto estas entradas son necesarias. También se puede integrar otro tipo de sensor de tensión y corriente que utilice un protocolo de comunicación disponible (I^2C ó SPI) para liberar 2 entradas analógicas.

En cuanto a las salidas de señales PWM se requieren dos para controlar los servo actuadores de comando y una tercera que se podría utilizar para controlar un servo actuador de control de actitud (actualmente está ocupada en un control de acelerador en el prototipo de pruebas). En este caso todas las placas cumplen este requerimiento, teniendo todas posibilidad de ampliar la cantidad de salidas PWM para control de otros dispositivos.

Sería deseable tener disponible dos o más pines digitales para uso general que se puedan ocupar de acuerdo a los requerimientos particulares que surjan en el desarrollo del módulo de comando. Para este requerimiento, todas las placas excepto el modelo Arduino M0, tienen disponibles pines para ser utilizados con esta funcionalidad.

2.4.2 Interfaces de comunicación

Todos los sensores que se utilizan para el control y la navegación del paracaídas se encuentran conectados mediante alguna de las siguientes tres interfaces de comunicación: USART, I^2C ó SPI. En cuanto a la disponibilidad de estas interfaces todos los micro controladores de las placas de desarrollo seleccionadas tienen soporte para las mismas a nivel de hardware, sólo queda analizar si la cantidad de estas interfaces mapeadas a pines de las placas de desarrollo es suficiente.

La interfaz de comunicación UART requiere dos pines digitales por periférico conectado, siendo los dispositivos / servicios que la utilizan los siguientes:

- Sistema de posicionamiento GPS
- Sistema de comunicación inalámbrica (telemetría)
- Consola de configuración

En total se requieren tres interfaces UART que ocupan 6 pines digitales en total de las placas de desarrollo (2 por dispositivo/servicio). La placa ArduinoMega dispone de 4 interfaces UART TTL de 5 V conectadas a pines específicos de la misma; la placa Arduino M0 también tiene 1 interfaz UART asignada a pines específicos y 3 interfaces USART TTL de 3,3V que puede ser mapeadas a cualquier pin digital de la placa [7]. La placa Teensy 3.6 tiene disponible 6 interfaces USART que pueden ser mapeadas a cualquier pin digital. Finalmente la placa LoPy 4 dispone de 3 interfaces UART que pueden ser mapeadas a cualquier pin digital. Como conclusión todas las placas cumplen el requerimiento de disponer 3 interfaces USART para los servicios / dispositivos mencionados.

La interfaz I^2C requiere de dos pines digitales pero en los mismos se pueden conectar hasta 127 dispositivos (asumiendo que las direcciones de los mismos se representan como un entero de 7 bit). La misma es utilizada por los siguientes dispositivos / servicios:

- Sistema de medición de condiciones ambientales (barómetro)
- Sistema de medición de rumbo (magnetómetro)
- Sistema de medición de altura para fase final de aterrizaje (altímetro laser)

Este requerimiento lo cumplen todas las placas ya que todas disponen de una interfaz I^2C; para la misma sólo se ocupan dos pines digitales para las señales SDA Y SCL del protocolo, estando todos los dispositivos conectados al mismo bus.

La interfaz SPI utiliza 3 pines comunes a los que se conectan todos los dispositivos (MISO, MOSI y SCK), mientras que cada dispositivo individual debe estar conectado a un pin digital individual denominado CS que se utiliza para habilitar la comunicación entre el periférico y la computadora principal. La misma es utilizada por los siguientes dispositivos / servicios:

- Registro de datos abordo del módulo

Al igual que antes todas las placas disponen de una interfaz de comunicación SPI y se podría conectar al menos un dispositivo que use este protocolo. La utilización de más periféricos estará sujeta a la cantidad de pines digitales libres (y si es posible mapearlos) para usarlos como pin CS.

2.4.3 Listado de pines

Usando los diagramas de mapeo de pines [8] [9] [10] [11] de las placas de desarrollo seleccionadas, se generó la Tabla 6, a continuación, para corroborar que todos los pines pueden ser utilizados para las funcionalidades requeridas y no exista superposición de los mismos. La nomenclatura utilizada para designar los pines es la que figura en los diagramas de mapeo de las referencias.

Dispositivo / servicio	Interfaz / tipo de pin	ArduinoMega pin #	Arduino M0 pin #	Teensy 3.6 pin #	LoPy 4 pin #
Entrada pwm motor	Entrada analógica #1	A0 (ADC0)	A0 (AIN0)	A17 (36)	ADC1_0 (5)
Entrada pwm comando 1	Entrada analógica #2	A1 (ADC1)	A1 (AIN2)	A18 (37)	ADC2_6 (17)
Entrada pwm comando 2	Entrada analógica #3	A2 (ADC2)	A2 (AIN3)	A19 (38)	ADC1_2 (7)
Entrada pwm modo de vuelo	Entrada analógica #4	A3 (ADC3)	A3 (AIN4)	A20 (39)	ADC1_3 (8)
Entrada pwm cambio modo	Entrada analógica #5	A4 (ADC4)	A4 (AIN5)	A7 (21)	ADC1_7 (11)
Tensión de batería	Entrada analógica #6	A5 (ADC5)	A5 (AIN10)	A8 (22)	ADC1_6 (10)
Consumo de corriente	Entrada analógica #7	A6 (ADC6)	No disponible	A9 (23)	ADC1_4 (12)
Salida pwm actuador 1	Salida pwm #1	D6	D7 (GPIO1)	PWM (2)	GPIO_33 (13)
Salida pwm actuador 2	Salida pwm #2	D7	D8 (GPIO2)	PWM (6)	GPIO_26 (15)
Salida pwm actuador 3	Salida pwm #3	D8	D9 (GPIO3)	PWM (35)	GPIO_25 (14)
Consola	UART	D0 (RX0) D1 (TX0)	D0 (PA11) D1 (PA10)	RX1 (0) TX1 (1)	RX0 (40) TX0 (41)
Telemetría	UART	D19 (RX1) D18 (TX1)	D2 (PA11) D3 (PA10)	RX2 (9) TX2 (10)	RX1 (21) TX1 (24)
GPS	UART	D17 (RX2) D16 (TX2)	D4 (PA14) D5 (PA15)	RX3 (7) TX3 (8)	No disponible
Bus I^2C (varios servicios)	I^2C	D20 (SDA) D21 (SCL)	D12 (SDA) D13 (SCL)	SDA0 (18) SCL0 (19)	SDA (18) SCL (20)
Registro de datos	SPI	D50 (MISO) D51 (MOSI) D52 (SCK) D53 (SS)	ICSP1 (MISO) ICSP4 (MOSI) ICSP3 (SCK) D11 (SS)	MISO0 (12) MOSI0 (11) SCK0 (14) CS0 (15)	MISO (6) MOSI (39) GPIO_2 (22) GPIO_0 (23)

Tabla 6: Listado de pines a utilizar por placa de desarrollo

Las placas de desarrollo Arduino Mega y Teensy 3.6 satisfacen los requerimientos en cuanto a disponibilidad y funcionalidades de los pines para los servicios del módulo de comando de paracaídas; además, ambas placas tienen posiblidades de amplicación ya que hay disponibles pines libres. En particular la placa Teensy 3.6 cuenta con mejores capacidades de ampliación al quedar libres la siguiente cantidad de interfaces: 3 USART, 3 I^2C, 1 SPI, 2 CAN. En cambio las placas Arduino M0 y LoPy4, satisfacen la funcionalidad de los pines pero no la disponibilidad de los mismos para el requerimiento actual. Relajando el requerimiento y eliminando algunos servicios que no serían necesarios en el producto final del módulo de comando, se satisfacería la disponibilidad de pines y ambas placas podrían ser usadas.

2.4.4 Migración del programa de autopiloto

A continuación se analiza brevemente la factibilidad de migrar el programa existente de autopiloto para entrega de carga mediante paracaídas a las diferentes placas de desarrollo seleccionadas. Para el mismo se asume que no existen problemas de compatibilidad con los módulos de Arduino. Tres de las cuatro placas seleccionadas utilizan una arquitectura de micro controlador diferente a la del hardware disponible actualmente y sobre el que se ha realizado todo el desarrollo del código de autopiloto. Por lo tanto, de acuerdo con la Figura 1, la porción de código que depende directamente de las librerías de AVR debería reescribirse adecuando las funcionalidades a la librería del microcontrolador de cada placa de desarrollo.

Para tener una idea del impacto de la falta de compatibilidad con las librería de AVR, se listaron los archivos del programa de autopiloto y de las librerías que dependen directamente de los archivos de la librería de AVR (ver Tabla 7 y Tabla 8). Los módulos que están construidos a partir de estos archivos requerirían una revisión para adaptarlos a las funcionalidades particulares del micro controlador elegido.

2.4.4.1 Estadística de dependencia del programa y librerías del autopiloto

El programa de autopiloto depende directamente de 2 archivos de la librería AVR: <avr/eeprom.h> y <avr/pgmspace>. Siendo el número total de archivos que forma el código del autopiloto de 90 (entre archivos de encabezado y archivos de código fuente), los porcentajes de archivos que dependen de la librería <avr/eeprom.h> es del 14,4 %, mientras que el porcentaje que depende de la librería <avr/pgmspace> es del 12,2 %. Estos porcentajes son relativamente bajos, sin embargo, habría que realizar un análisis de la profundidad de las modificaciones en el código fuente que habría que introducir (puede ser el caso en que solamente halla que realizar un cambio por una función alternativa para el micro controlador elegido o que halla que reescribir completamente una parte del archivo ya que no se dispone de una función equivalente).

En el caso de las librerías, las mismas depende directamente de 4 archivos de AVR: <avr/eeprom.h>, <avr/interrupt>, <avr/io> y <avr/pgmspace>. En este caso la cantidad total de archivos es de 99, por lo que los porcentajes de dependencia con módulos de la librería de AVR son los siguientes: el porcentaje de archivos que dependen de la librería <avr/eeprom.h> es del 1 %, el porcentaje que depende de la librería <avr/interrupt> es del 5 %, el porcentaje que depende de la librería <avr/io> es del 2 %, y finalmente el porcentaje que depende de la librería <avr/pgmspace> es del 4 %. Estos porcentajes son menores con respecto al programa de autopiloto pero afectan a algunos módulos que son utilizados por muchos sistemas del autopiloto como ser la clase FastSerial.

Archivo encabezado AVR	Archivos del programa de autopiloto
<avr/eeprom.h>	EEPROM.cpp adsControlNavigation.cpp adsGuidance.cpp adsModes.cpp adsSystem.cpp controllers/adsControllers.cpp controllers/adsLQHeadController.cpp controllers/adsMRASHeadControoler.cpp controllers/adsPIDHeadController.cpp ioControlPWM.cpp logSystem.cpp navigation/adsNavigationSystem.cpp setup.cpp
<avr/pgmspace>	adsSystem.cpp auxFunctions.cpp controllers/adsControllers.cpp controllers/adsLQHeadController.cpp controllers/adsMRASHeadControoler.cpp controllers/adsPIDHeadController.cpp controllers/controllersGlobalData.cpp globalVar.cpp navigation/adsNavigationSystem.cpp setup.cpp system.cpp

Tabla 7: Archivos del programa de autopiloto que dependen directamente de un archivo de la librería AVR

Archivo encabezado AVR	Archivos de las librerías del autopiloto
<avr/eeprom.h>	AP_Common/AP_Var.h
<avr/interrupt.h>	APM_BMP085/APM_BMP085.cpp APM_RC/APM_RC.cpp AP_ADC/AP_ADC_ADS7844.cpp DataFlash/DataFlash.cpp FastSerial/FastSerial.h
<avr/io.h>	AP_Common/AP_MetaClass.h FastSerial/FastSerial.h
<avr/pgmspace>	AP_Common/AP_Var.h AP_Common/menu.cpp FastSerial/BetterStream.h FastSerial/vprintf.cpp

Tabla 8: Archivos de las librerías del autopiloto que dependen directamente de un archivo de la librería AVR

2.4.4.2 Migración a ArduinoMega

Esta placa de desarrollo tiene el mismo micro controlador que la placa de autopiloto que se está utilizando por lo que no es necesario realizar una migración de código. El software del autopiloto se puede embeber directamente en el micro controlador. Solamente sería necesario hacer una re-asignación de pines de acuerdo a la distribución de la placa de desarrollo (la propuesta de la Tabla 6 puede servir de guía inicial para esta asignación).

2.4.4.3 Migración a Arduino M0

Esta placa de desarrollo utiliza un micro controlador con arquitectura ARM Cortex, por lo que el compilador y las librerías del mismo son diferentes a las que se están utilizando actualmente. El entorno de programación de Arduino provee algunos reemplazos para poder compilar código generado para arquitecturas de AVR usando la placa Arduino M0; en la Tabla 9 se muestra un listado de las librerías de AVR que tienen compatibilidad para esta placa. La misma se generó usando el entorno de programación Arduino 1.8.5 y el paquete de librería y compilador arm-none-eabi versión 5.4.1.

Archivo de encabezado	Descripción	Soporte para el compilador ARM
<avr/boot.h>	Herramientas de soporte del bootloader	NO
<avr/cpufunc.h>	Funciones especiales para el CPU AVR	NO
<avr/eeprom.h>	Manejo de la memoria EEPROM	NO
<avr/fuse.h>	Soporte de fusibles	NO
<avr/interrupt.h>	Interrupciones	SI
<avr/io.h>	Definiciones de IO específicas para dispositivos de AVR	SI
<avr/lock.h>	Soporte de lockbit	NO
<avr/pgmspace.h>	Utilidades del espacio de programa	SI
<avr/power.h>	Administración para la reducción del consumo	NO
<avr/sfr_defs.h>	Funciones especiales de registros	NO
<avr/signature.h>	Soporte de firma	NO
<avr/sleep.h>	Administración de consumo y modos de espera (sleep)	NO
<avr/version.h>	Macros de la versión de la librería avr-libc	NO
<avr/wdt.h>	Administración del reloj del watchdog	NO

Tabla 9: Listado de librerías de AVR que se pueden compilar usando las librerías y el compilador para la placa de desarrollo Arduino M0

La falta de compatibilidad se encuentra en la librería que administra la lectura y escritura en la memoria permanente del micro controlador. Seguramente las secciones de código que se refieren al almacenamientos de datos en la memoria EEPROM se puede reemplazar por funciones específicas de la librería de ARM.

2.4.4.4 Migración a Teensy 3.6

Esta placa de desarrollo no presenta problemas de compatibilidad de nivel de las librerías de AVR, ya que tiene implementadas versiones alternativas para los módulos que utiliza el programa de autopiloto, tal como se muestra en la Tabla 10. Sin embargo, existe un problema de compilación con la librería FastSerial que reemplaza a la librería Serial de Arduino. Hay que investigar si este inconveniente se puede corregir mediante un orden especial de compilación en el archivo Makefile ó si es necesario eliminar el uso de la librería FastSerial del programa de autopiloto y reemplazarla por las librerías de puerto serie provistas para esta placa de desarrollo.

Archivo de encabezado	Descripción	Soporte para el compilador ARM
<avr/boot.h>	Herramientas de soporte del bootloader	NO
<avr/cpufunc.h>	Funciones especiales para el CPU AVR	NO
<avr/eeprom.h>	Manejo de la memoria EEPROM	SI
<avr/fuse.h>	Soporte de fusibles	NO
<avr/interrupt.h>	Interrupciones	SI
<avr/io.h>	Definiciones de IO específicas para dispositivos de AVR	SI
<avr/lock.h>	Soporte de lockbit	NO
<avr/pgmspace.h>	Utilidades del espacio de programa	SI
<avr/power.h>	Administración para la reducción del consumo	SI
<avr/sfr_defs.h>	Funciones especiales de registros	NO
<avr/signature.h>	Soporte de firma	NO
<avr/sleep.h>	Administración de consumo y modos de espera (sleep)	SI
<avr/version.h>	Macros de la versión de la librería avr-libc	NO
<avr/wdt.h>	Administración del reloj del watchdog	SI

Tabla 10: Listado de librerías de AVR que se pueden compilar usando las librerías y el compilador para la placa de desarrollo Teensy 3.6

2.4.4.5 Migración a LoPy 4

Finalmente, para esta placa se probó la compilación usando la librería de Arduino adaptada para el MCU ESP32 [12] provista por el fabricante del mismo (Espressif). En este caso no hay soporte para las 4 librerías de AVR que utiliza el programa de autopiloto (ver Tabla 11), por lo que una migración a este hardware requeriría mayor cantidad de revisión del código para tener una versión operativa.

Archivo de encabezado	Descripción	Soporte para el compilador ARM
<avr/boot.h>	Herramientas de soporte del bootloader	NO
<avr/cpufunc.h>	Funciones especiales para el CPU AVR	NO
<avr/eeprom.h>	Manejo de la memoria EEPROM	NO
<avr/fuse.h>	Soporte de fusibles	NO
<avr/interrupt.h>	Interrupciones	NO
<avr/io.h>	Definiciones de IO específicas para dispositivos de AVR	NO
<avr/lock.h>	Soporte de lockbit	NO
<avr/pgmspace.h>	Utilidades del espacio de programa	NO
<avr/power.h>	Administración para la reducción del consumo	NO
<avr/sfr_defs.h>	Funciones especiales de registros	NO
<avr/signature.h>	Soporte de firma	NO
<avr/sleep.h>	Administración de consumo y modos de espera (sleep)	NO
<avr/version.h>	Macros de la version de la librería avr-libc	NO
<avr/wdt.h>	Administración del reloj del watchdog	NO

Tabla 11: Listado de librerías de AVR que se pueden compilar usando las librerías y el compilador para la placa de desarrollo LoPy 4

3. CONCLUSIONES

De la revisión realizada en las secciones anteriores se determinó que la placa de desarrollo que presenta el mejor balance entre ampliación de las capacidades del hardware disponible actualmente y esfuerzo para migrar el código es el modelo Teensy 3.6 ya que la misma dispone de un micro controlador más potente que la placa de autopiloto disponible actualmente, además de ampliar la cantidad de periféricos disponibles para conectar sensores y sistemas auxiliares a la misma y es compatible con las librerías de AVR y Arduino que están siendo utilizadas en el software de control.

Existe un inconveniente con la compilación de la librería FastSerial que reemplaza a la librería estándar Serial para el manejo de periféricos conectados a un puerto UART del microcontrolador, que hay que investigar en detalle para determinar si se trata de un problema de compatibilidad de la librería o solamente de un problema de orden de compilación.

4. REFERENCIAS

[1] REYNOSO, S. *"Descripción del hardware del autopiloto ArduPilotMega"*. Dpto de Mecánica Aeronáutica, Facultad de Ingeniería, Instituto Universitario Aeronáutico. Informe técnico DMA-017/11. Noviembre de 2011.

[2] LLORENS, D. *"Descripción del código de autopiloto para paracaídas comandado autónomo"*. Dpto de Mecánica Aeronáutica, Facultad de Ingeniería, Instituto Universitario Aeronáutico. Informe técnico DMA-014/17. Septiembre de 2017.

[3] Ardupilot Dev Team. *Autopilot Hardware Options – Plane documentation* [en línea]. [s/d: Ardupilot Dev Team], s/d., s/d. <http://ardupilot.org/plane/docs/common-autopilots.html> [Consulta: 26 de Marzo de 2018]

[4] Ardupilot Dev Team. Pixhawk – *Plane documentation* [en línea]. [s/d: Ardupilot Dev Team], s/d., s/d. <http://ardupilot.org/plane/docs/common-pixhawk-overview.html> [Consulta: 26 de Marzo de 2018]

[5] Arduino AG. Arduino *Reference* [en línea]. [USA: Arduino AG], s/d., s/d. <https://www.arduino.cc/reference/en/> [Consulta: 3 de Abril de 2018]

[6] Atmel. *Modules – AVR Libc Reference Manual* [en línea]. [USA: Microchip Technology Inc.], s/d., s/d. <https://www.microchip.com/webdoc/AVRLibcReferenceManual/ch20.html> [Consulta: 3 de Abril de 2018]

[7] Arduino AG. *Arduino – SamdSercom: Adding more Serial Interfaces to SAMD microcontrollers (SERCOM)* [en línea]. [USA: Arduino AG], s/d., s/d. <https://www.arduino.cc/en/Tutorial/SamdSercom> [Consulta: 3 de Abril de 2018]

[8] Arduino AG. *Arduino - PinMapping2560* [en línea]. [USA: Arduino AG], s/d., s/d. <https://www.arduino.cc/en/Hacking/PinMapping2560> [Consulta: 5 de Abril de 2018]

[9] Arduino AG. *Arduino-M0-V1.sch* [en línea]. [USA: Arduino AG], s/d., s/d. <https://www.arduino.cc/en/uploads/Main/arduino-M0-schematic.pdf> [Consulta: 5 de Abril de 2018]

[10] PJRC Electronic projects. *Teensy and Teensy++ Pinouts, for C language and Arduino Software* [en línea]. [Oregon, USA: PJRC.COM], s/d., s/d. <https://www.pjrc.com/teensy/pinout.html> [Consulta: 5 de Abril de 2018]

[11] Pycom. *Pycom_Specsheet_LoPy4.pdf* [en línea]. [Surrey, United Kingdom: Pycom.io], s/d., s/d. <https://pycom.io/wp-content/uploads/2018/03/Pycom_Specsheet_LoPy4.pdf> [Consulta: 5 de Abril de 2018]

[12] Espressif Systems. *Arduino-esp32:Arduino core for ESP32 WiFi chip* [en línea]. [USA: GitHub], s/d., s/d. <https://github.com/espressif/arduino-esp32> [Consulta: 5 de Abril de 2018]