Entrega final del proyecto: Sistema automatizado de alimentación para mascotas.

Santiago A. Campos Mora[1], Juan Pablo Prieto Vergara [2],
Santiago Herrera Acuña[3],
Electrónica Digital II (2016499), Departamento de Eléctrica y Electrónica
Facultad de Ingeniería
Universidad Nacional de Colombia, Sede Bogotá
[1]scamposm@unal.edu.co, [2]jprietov@unal.edu.co, [3]saherreraa@unal.edu.co

***Abstract*— The project aims to develop an automated pet feeding system controlled by a Zybo Z7 FPGA. This device seeks to ensure the delivery of controlled portions of food to our pets at set times, thus contributing to their well-being and proper nutrition.**

It integrates an efficient mechanical design manufactured using 3D printing, as well as a digital control system based on Verilog modules and wireless communication via Bluetooth (UART communication). Through a mobile application developed for Android devices, the user can remotely manage their pet's feeding.

The project presents a low-cost alternative to commercial dispensers, with the potential to become a student-led technological entrepreneurship product.

***Index Terms*—PESTAL, mascota, impresión 3d, automático, comunicación serial UART, sistema digital.**

I. Resumen

El proyecto tiene como objetivo el desarrollo de un sistema automatizado de alimentación para mascotas, controlado mediante una FPGA Zybo Z7, este dispositivo busca asegurar la entrega de porciones controladas de alimento para nuestras mascotas en horarios establecidos, con esto contribuyendo al bienestar y la correcta nutrición de nuestros peludos.

El integra un diseño mecánico eficiente fabricado con impresión 3D ademas un sistema de control digital basado en codigo en C por medio de VITIS y VIVADO (con un block desing) y comunicación inalámbrica mediante Bluetooth(comunicación Uart). A través de una aplicación móvil desarrollada para dispositivos Android, el usuario podrá gestionar la alimentación de su mascota de forma remota.

el proyecto plantea una alternativa de bajo costo frente a los dispensadores comerciales, con potencial de convertirse en un producto de emprendimiento tecnológico estudiantil.

II. Introducción

En la actualidad, la automatización y el uso de tecnologías inteligentes se han extendido a diferentes ámbitos de la vida cotidiana, incluyendo el cuidado de los animales domésticos, en este ambito uno de los problemas más comunes para los dueños de mascotas es la ïmposibilidad"de mantener horarios regulares de alimentación ya sea a compromisos laborales, académicos y/o personales, esta situación de no tener horario regular puede afectar la salud y el comportamiento de los animales, generando ansiedad, sobrepeso o desnutrición.

Con esto en mente, surge la idea de desarrollar un sistema que facilite la alimentación automática y programada de las mascotas, por esto el presente proyecto propone un dispensador automatizado de alimento que permita controlar el momento de distribución, empleando como núcleo de procesamiento la FPGA Zybo Z7.

III. Objetivos

Objetivo principal: Diseñar un sistema automatizado que permita asegurar la dispersión de porciones controladas de alimento para mascotas, haciendo uso de la FPGA Zybo Z7 y su procesador ZYNQ como principal elemento.

Objetivos secundarios:

- Diseñar un prototipo mecánicamente eficiente que permita dispensar el alimento y almacenarlo por un periodo de tiempo razonable y cómodo para el usuario.
- Controlar el sistema a distancia por medio de una aplicación de autoría propia descargable en cualquier dispositivo con sistema operativo Android.
- Implementar un segundo elemento que permita la activación del sistema por parte de la mascota, en caso tal de que el propietario no se encuentre en el hogar.

IV. Identificación del problema

La mayoría de dueños de mascotas no siempre tienen disponibilidad de tiempo para alimentar a sus animales en los horarios regulares debido a compromisos, estudios u otras actividades de la vida diaria, además esta irregularidad en la alimentación afecta la salud de nuestras mascotas, ya que pueden quedar sin alimento por varias horas o también pueden recibir una cantidad inadecuada o ser sobrealimentadas en un solo momento del día. Estos desequilibrios pueden provocar problemas de comportamiento como ansiedad, estrés o hiperactividad, y en el aspecto físico, sobrepeso, desnutrición y alteraciones metabólicas (Coulthard et al., 2017; Mistlberger, 2011).

Con esto en mente, se tiene la idea de implementar con soluciones tecnológicas que permitan automatizar la dispensación de comida, con esto asegurando raciones controladas y algo importante, que es en los horarios debidos y constantes, por esto el uso de dispositivos inteligentes conectados a

aplicaciones da la posibilidad de gestionar la alimentación de sus mascotas a distancia, dando asi mayor tranquilidad y garantizando un cuidado más responsable y adaptado a las necesidades de cada animal (Coherent Market Insights, 2023; Global Market Insights, 2023).

V. Análisis PESTAL

Político: Este proyecto se puede implementar ya que no existen barreras políticas relevantes para el proyecto, salvo regulaciones de conectividad inalámbrica, además se puede apoyar en políticas y programas de innovación tecnológica que brinda la universidad.

Económico: El prototipo representa una alternativa que puede resultar de bajo costo frente a productos importados del mercado de dispensadores automáticos que generalmente son costosos y poco accesibles para el sueldo de estudiantes. La Universidad Nacional de Colombia puede destinar recursos de semilleros de investigación, para financiar parcialmente su desarrollo. A futuro este proyecto puede convertirse en una oportunidad de emprendimiento estudiantil.

Social: El proyecto contribuye a mejorar el bienestar de las mascotas de estudiantes y docentes que, por sus horarios académicos y laborales, no siempre tienen tiempo de alimentar a sus animales en momentos adecuados.

Tecnológico: El proyecto connota grandes niveles de competencia, ya que el producto se encuentra en el país, con empresas como Volutas, BE, Xiaomi, VTA, entre otros. Presentan grandes avances en cuanto a la conexión con aplicaciones, uso de transmisión de datos y seguimiento real por medio de internet de gran alcance inalámbrico, y tecnologías IoT. Sin embargo, hasta la fecha, no presentan implementaciones de Inteligencia Artificial en sus sistemas.

Ambiental: Se presume un sistema duradero al seguir las instrucciones del fabricante, por lo que estos dispositivos suelen tener larga vida útil. Es usual que estos productos añadan en su diseño materiales como cartuchos descartables, que al momento de absorber humedad de un alimento deben ser tratados como residuo, ya que luego de un mes después de la apertura se recomienda su reemplazo. También, como alternativa a la corriente en caso de un fallo de la red, algunos de estos dispensadores funcionan a base de baterías, siendo un potencial residuo contaminante por contener metales pesados y sustancias tóxicas. Por último, para la estructura física, es común el uso de plástico como carcasa, el cual es altamente contaminante.

Legal: Al ser objetivo del proyecto la extensión del cuidador para alimentar a su mascota, y en consecuencia, una facilidad para el mismo, la situación se encuentra en una línea delgada sobre la omisión de responsabilidades del cuidador. Este se vería en un área aún más delicada bajo procesos de automatización o desarrollo de tecnología de inteligencia artificial, bajo la cual, un fallo en las debidas proporciones desencadena en un problema legal. Bajo el Estatuto Nacional de Protección Animal, ley 1774 de 2026, artículo 3, sección b1, “Los animales no deben sufrir, ni hambre, ni sed”, por lo que, cualquier falla tanto en la programación del dispositivo como en fallas de conexión, tanto a la red por alimentación, como del enlace de comunicación, puede desencadenar en consecuencias legales para el cuidador o la desarrolladora del producto.

VI. Antecedentes

Comederos de mascotas existentes: Los comederos de mascotas ya se encuentran establecidos en los mercados bajo empresas especializadas en mascotas (como Pawaboo) o empresas centradas en tecnologías IoT (Xiaomi). Pero, adicional a esto, también es considerable el sector de proyectos independientes que usan tecnologías de microcontroladores para el desarrollo de prototipos. Entre ellos se encuentran:

- Microcontrolador ESP8266 controlable mediante Telegram.
- Servomotores controlados por Arduino de software abierto.
- Combinación de IoT junto a aplicaciones móviles para gestión de horarios (patente Olsson).

VII. Posibles soluciones al problema

Para resolver el problema que abarca este proyecto, se estiman algunas soluciones las cuales pueden o no alinearse con los objetivos de esta materia. Es probable que todas las soluciones propuestas en este documento no se lleven a cabo, sin embargo, se buscará como prioridad dar solución usando los conocimientos adquiridos en la clase y en el ámbito de la ingeniería electrónica. Existen varias soluciones para este problema, entre ellas está generar una manera de avisar/recordar al propietario de la mascota que es hora de llenar el recipiente con alimento, debido a un ritmo de trabajo demandante que esta persona pueda tener. Otra solución es gestionar una red de cuidadores al servicio del propietario que se encargue de alimentar y cuidar a la mascota el tiempo que se pacte. Así mismo, se pueden ofrecer servicios de cuidado, como una guardería especializada que quite preocupaciones al dueño y ofrezca servicios adicionales como entrenamiento. Por supuesto, las soluciones propuestas anteriormente, que aunque útiles, no se alinean con el objetivo de la clase ni de la carrera, por ese motivo, las soluciones propuestas para esta problemática se abarcaran desde el ámbito de la ingeniería electrónica digital. Una solución propuesta sería crear un sistema automático que facilita la alimentación diaria de la mascota, acomodándose a la rutina del propietario .

VIII. Actores de la solución y sus roles

Santiago Alejandro Campos Mora: Programador de Hardware en la FPGA. Su rol en la solución del problema será administrar la FPGA y generar los módulos individuales para la comunicación con cada sensor y actuador presente. Además, elaborará diagramas de flujo, máquinas de estado y planteará soluciones lógicas para el desarrollo en hardware.

Juan Pablo Prieto Vergara: Diseñador de la parte mecánica del sistema. Su rol en la solución del problema será diseñar cada pieza física del sistema para posteriormente imprimirla y/o cortarla y luego ensamblarla en un todo, buscando optimizar los recursos y el espacio del prototipo.

Santiago Herrera Acuña: Su rol en la solución del problema será unir y supervisar cada una de las partes. En la programación de hardware, creará el módulo `.Top` y unirá cada módulo individual, depurando código en la marcha y enlazando las partes. En la parte mecánica, revisará el diseño y se asegurará de que las piezas no presenten errores y se puedan unir correctamente. Posteriormente, se encargará de decidir qué material será el más óptimo a usar.

IX. Presupuesto

Elemento	Costo $(COP)
Piezas mecánicas impresas en 3D	$\approx$ $40.000
Módulo Bluetooth HC-06	$\approx$ $17.000
Sensor de proximidad Radar Doppler RCWL-0516	$\approx$ $7.000
Servo Motor	$\approx$ $15.000
Relés (asumiendo más de 2 unidades)	$\approx$ $7.000
Bornera de 2 pines (asumiendo más de 2 unidades)	$\approx$ $3.000
Resistencias varias	$\approx$ $2.000
Cables de conexión	$\approx$ $2.000
Tornillos	$\approx$ $5.000
PCB	$\approx$ $6.000
Mano de obra	$\approx$ $170.000
Total	$\approx$ **$274.000**

Cuadro I: Presupuesto aproximado para la realización del proyecto:

X. Implementación del proyecto

X-A. *Diseño del prototipo mecánico*

Uno de los objetivos clave para este proyecto se basa en diseñar un prototipo solido y de tamaño razonable. Debido a que se busca versatilidad a la hora de transportarse o cambiarse de lugar, el prototipo no solo debería ser estable y resistente, si no también de tamaño mediano y ligero. Se opto entonces por realizar el prototipo con impresión 3D aprovechando el fácil acceso que concede la universidad a este servicio.

No solo se contaba con impresoras modernas, si no que la persona encargada podía asesorar el proceso para que cada pieza quedase con el mejor acabado. Para el inicio de dicho proceso fue necesario realizar la compra del material que se usaría para la impresión, en este caso se opto por el uso de PLA de color negro. Luego de varios bocetos (cada uno de ellos descartado) se buscaron en linea referencias de diseños ya implementados, uno de ellos fue el tornillo sin fin que resultaba ser a primera vista una opción atractiva, pero indagando mas a fondo, este diseño presentaba fallas a la hora de usarse debido a que la comida independientemente de su tamaño atascaba el mecanismo del tornillo y ni siquiera la fuerza del motor podía destrabarlo. Al final, se encontró un diseño que parecía funcionar basado en que la comida cayera por acción de la gravedad y su propio peso, y cuyo mecanismo es una trampilla móvil controlada con un servo motor.

Basado en este diseño y haciendo uso de Tinkercad, se modificaron las piezas necesarias y se imprimió cada una de ellas haciendo uso del servicio de impresoras de la universidad.

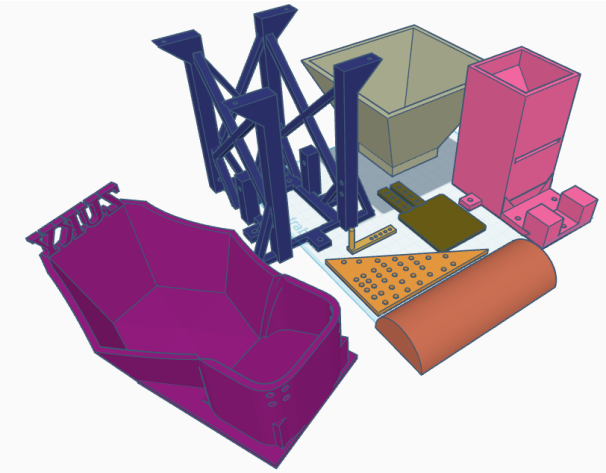

Figura 1: Piezas diseñadas en Tinkercad

Según el presupuesto estimado, la impresión costaría $40.000 pesos, sin embargo, el precio final fue de $70.000. En total, las piezas estuvieron listas después de 3 días de impresión, debido a que algunas (las mas grandes) presentaron problemas, sumado a que el servicio de impresión es altamente demandado, por lo cual se debía realizar una reserva de las impresoras siendo 4 horas de impresión el tiempo máximo permitido por día. La base azul mostrada en la figura 1 fue la mas problemática, pues tardaba tres horas en imprimirse en la mejor impresora pero fallaba faltando 10 minutos de impresión. En el anexo de este informe se encuentran las evidencias fotográficas de cada una de las piezas.

Una vez impresas las piezas se unieron mecánicamente por medio de tornillos y pegamento, obteniendo el resultado de la figura 2.

Figura 2: Vista general del prototipo

X-B. Perifericos y componentes

X-B1. Sensor RCWL-0516

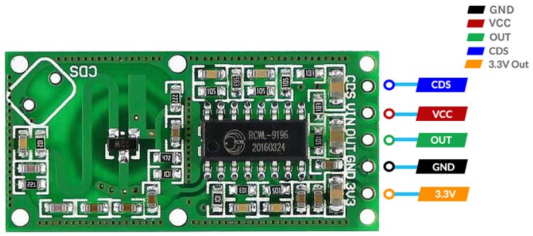

Figura 3: sensor RCWL-0516

El **RCWL-0516** es un sensor de microondas basado en efecto Doppler, utilizado para detectar movimiento sin necesidad de contacto o luz visible. Opera a una frecuencia de aproximadamente **3.2 GHz** y es capaz de detectar objetos en movimiento en un rango típico de **5 a 7 metros**. A diferencia de los sensores PIR, el RCWL-0516 puede detectar movimiento incluso a través de ciertos materiales como plástico, vidrio o madera delgada. Su salida es una señal digital que se activa cuando se detecta movimiento, lo que lo hace ideal para aplicaciones de seguridad, iluminación automática y sistemas embebidos.

X-B2. Modulo HC-06

Es un módulo electrónico diseñado para establecer comunicación inalámbrica de corto alcance mediante el protocolo Bluetooth. Su función principal es actuar como un puente serial (UART) transparente, permitiendo enviar y recibir datos sin cables entre un microcontrolador (como Arduino o PIC) y dispositivos maestros como teléfonos inteligentes o computadoras.

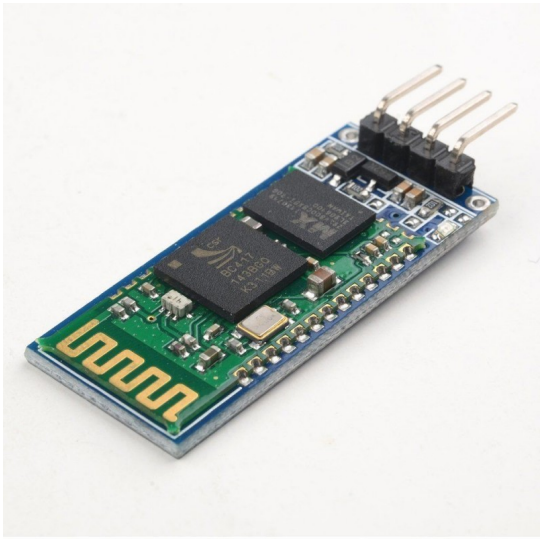

Figura 4: Modulo HC-06

X-B3. Servo motor

Figura 5: servomotor SG90

El **SG90** es un servomotor de pequeña escala ampliamente utilizado en proyectos de robótica y sistemas embebidos. Funciona con señal de control por PWM y permite posicionamiento angular típico entre **0° y 180°**. Opera con un voltaje de alimentación de **4.8 a 6 V** y ofrece un torque aproximado de **1.8 kg·cm**. Su tamaño compacto y bajo consumo lo hacen ideal para aplicaciones como control de pequeñas articulaciones, mecanismos de precisión y proyectos educativos.

X-B4. FPGA Zybo Z7

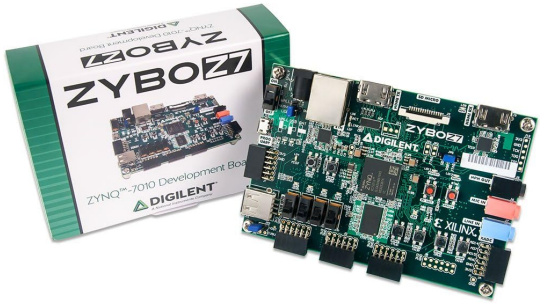

Figura 6: FPGA Zybo Z7

Es una plataforma de desarrollo de circuitos digitales y software embebido de alto rendimientoSe construye alrededor de la arquitectura Zynq-7000 AP SoC (System-on-Chip), que integra en un solo chip un procesador de doble núcleo (ARM Cortex-A9) junto con lógica programable de FPGA (basada en la serie Artix-7).Es ideal para proyectos de visión por computador, sistemas embebidos complejos y procesamiento de señales.

X-C. *Diseño de la aplicación para Android*

Para el diseño de la aplicación se utilizo la pagina web MIT app inventor. El objetivo de la app es conectarse por medio de bluetooth al modulo HC-06, de modo que este envie una señal

a la FPGA para activar el sensor. Se diseño bloque a bloque la aplicacion, cuya interfaz se evidencia a continuacion:

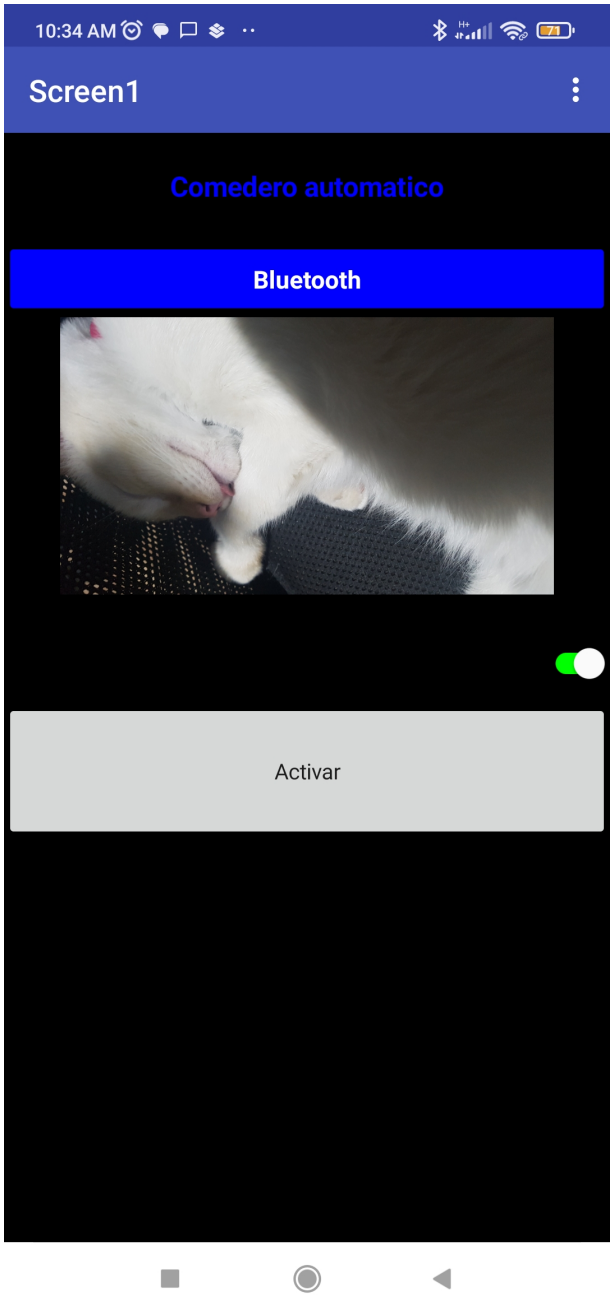

Figura 7: Vista de la app en un celular con sistema Andriod

Tal y como se observa en la figura 7, la interfaz de la aplicacion muestra un boton azul para conectarse a Bluetooth, una imagen de una de las gatas de los integrantes (Zuky) y un boton llamado Activar, el cual envia un caracter por medio de protocolo UART para activar el servo. El caracter que envia exactamente es el numero 1. Para programar este envio de informacion, en la propia aplicacion se diseñan bloques con la logica de envio y de bluetooth.

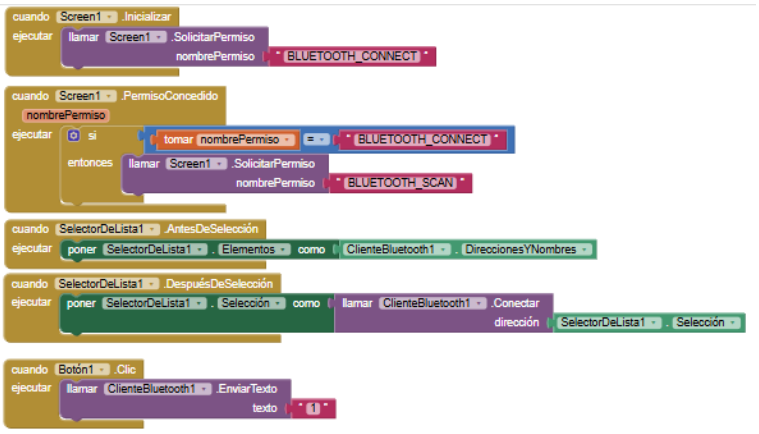

Figura 8: Bloques de programación para la app

la figura 8 permite observar la logica tras la aplicacion, la cual se realiza por medio de bloques muy sencillos que conectan el Bluetooth y que a su vez envian el caracter 1, el cual es recibido por el modulo HC 06 para enviar por UART a la FPGA.

X-D. Implementación en vivado del procesador

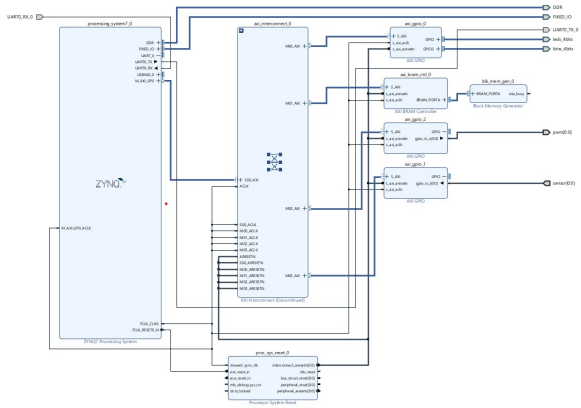

Figura 9: Vista general del procesador

El diseño mostrado corresponde a un sistema embebido basado en **Zynq-7000**, desarrollado en Vivado mediante la arquitectura **PS–PL**, donde el *Processing System (PS)* del Zynq interactúa con la lógica programable *Programmable Logic (PL)* a través del bus AXI.

X-D1. ZYNQ7 Processing System

El bloque principal del sistema es el **Processing System 7.0**, que integra:

- Procesador ARM Cortex-A9 de doble núcleo.
- Puertos **DDR** y **FIXED_IO** para comunicación con memoria externa.
- Interfaz **UART0** conectada a pines externos (RX y TX).
- Interfaz **M_AXI_GP0** empleada para acceder a periféricos AXI dentro de la PL.

El PS proporciona:

- Señal de reloj maestro **M_AXI_GP0_ACLK**.
- Señales de reinicio para la lógica programable.
- Comunicación con la matriz AXI mediante el puerto maestro.

X-D2. AXI Interconnect

El componente **axi_interconnect_0** actúa como un conmutador AXI que permite la comunicación entre el procesador ARM y múltiples periféricos en la PL. Posee:

- **S00_AXI**: conexión al PS (M_AXI_GP0).
- **M00_AXI, M01_AXI, M02_AXI, M03_AXI**: conexión hacia los periféricos AXI.

Este módulo se encarga del direccionamiento, arbitraje y señalización AXI4-Lite.

X-D3. Módulos AXI GPIO

El diseño contiene tres periféricos **AXI GPIO**, cada uno accesible desde el procesador mediante AXI4-Lite:

- **axi_gpio_0**: con dos canales GPIO (GPIO y GPIO2). Conectado a los puertos externos *leds_4bits* y *btns_4bits*.
- **axi_gpio_1**: canal de entrada, conectado a *sensor[0:0]*.
- **axi_gpio_2**: canal de salida, conectado al puerto *pwm[0:0]*.

Estos módulos permiten controlar LEDs, leer botones, manejar sensores digitales y generar PWM desde software.

X-D4. BRAM Controller y Block Memory Generator

El sistema incluye memoria interna implementada en la PL por medio de:

- **axi_bram_ctrl_0**: proporciona la interfaz AXI hacia el PS.
- **blk_mem_gen_0**: realiza la implementación física de la BRAM.

Esta memoria puede emplearse para almacenamiento temporal, buffers, tablas de valores o variables de tiempo real.

X-D5. Processor System Reset

El módulo **proc_sys_reset_0** genera señales de reinicio sincronizadas para:

- La red AXI.
- Los periféricos conectados.
- El sistema en general, usando el reloj del PS y la señal *FCLK_RESET0_N*.

Esto garantiza un arranque ordenado y estable del sistema embebido.

X-D6. Conexiones Externas

Entre las conexiones más relevantes del diseño se encuentran:

- Puertos **UART0_TX** y **UART0_RX** para comunicación serial con PC o microcontroladores.
- Señales **GPIO** conectadas a LEDs, botones, sensor digital y salida PWM.
- Todos los periféricos sincronizados mediante el reloj generado por el PS.

X-E. Programación en Vitis

Despues de creado el archivo .XSA en vivado por el bloc desing. se exporta este archivo a VITIS, con el objetivo de usar el archvio .Xparameters que contiene toda la informacion de los registros de nuestro proyecto. Con estas direcciones, creamos un codigo main.C el en donde usando lenguaje en C, programamos finalemte la logica detras de la activacion del servomotor, asi como el uso de los modulos HC-06 y el sensor RCWL-0516. Dicho codigo en C se encuentra en los anexos de este documento.

X-F. Video de la implementación

Se grabo un video mostrando el funcionamiento final del proyecto, en el explicamos brevemente los componentes usados y como funciona el dispensador. Este video se encuentra subido a YouTube de forma publica para que cualquier persona pueda acceder a el. El link del video es el siguiente:

- https://youtu.be/qVS-jUHCHdE

XI. Conclusiones

- La elección de un sistema automatizado de alimentación para mascotas como proyecto representa el desarrollo de una tecnología de alto interés social, con alto nivel de competitividad comercial, y que brinda muchas alternativas de desarrollo tecnológico en sistemas digitales.
- Se plantea un sistema de dispensación de alimento para mascota controlado a través de una aplicación desde la que se puede dispensar o bloquear la provisión de alimento como la mejor alternativa para mitigar la sobre-alimentación de la mascota.
- Se establece como prioridad el desarrollo de un enlace de comunicación digital, al ser el elemento central que permite reducir la exigencia al dueño de la mascota, en cuanto a su presencialidad.
- El sistema de comunicación UART, esencial para la conexión entre el dispositivo y el usuario, se muestra correctamente adaptable a las necesidades del proyecto por su fiabilidad de transmisión.
- Se concreta un diseño solido y a su vez soportable como prototipo mecánico basado en la acción de la gravedad para dispensar el alimento, debido a que de esta manera, se reducen los riesgos de obstrucción y sobre-esfuerzo del motor.
- Se plantea el sensor de proximidad como una señal de control de menor prioridad frente al uso del aplicativo.

Referencias

[1] A. Gabriel. (2019) Fpga04 implementar comunicación uart - camino a fpga. [Online]. Available: https://www.youtube.com/watch?v=xfjRfwK3hC4

[2] A. Rai. (2025) Pet tech market size and share analysis - growth trens and forecast (2025-2032). [Online]. Available: https://www.coherentmarketinsights.com/industry-reports/pet-tech-market

[3] E. García. (2024) Comedero y bebedero automatizado por internet para animales. [Online]. Available: https://www.youtube.com/watch?v=I5gg1VqX2v0

[4] J. G. M. Michelle Murphy, "Intermittent feeding schedules—behavioural consequences and potential clinical significance," *Nutrients*, vol. 6, no. 3, pp. 985–1002, 2014.

[5] V. Tripathi, S. K. Bhardwaj, and V. Kumar, "Ecology of timekeeping: feeding times effect clock-controlled behavior, metabolism and reproduction in diurnal vertebrates," *Biol Timing Sleep*, vol. 2, no. 8, 2025.

APÉNDICE A
ANEXO: PIEZAS IMPRESAS DEL PROTOTIPO

Figura 10: Base del prototipo

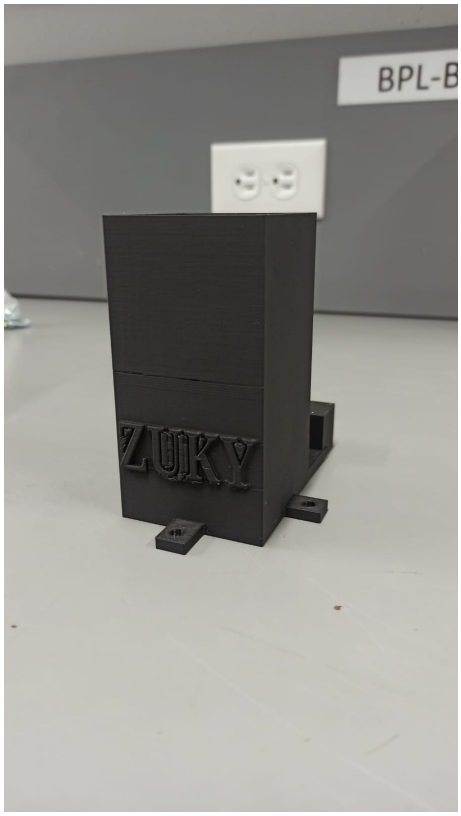

Figura 11: cuerpo del prototipo

Figura 12: Mecanismo del prototipo

Figura 13: Rampa del prototipo

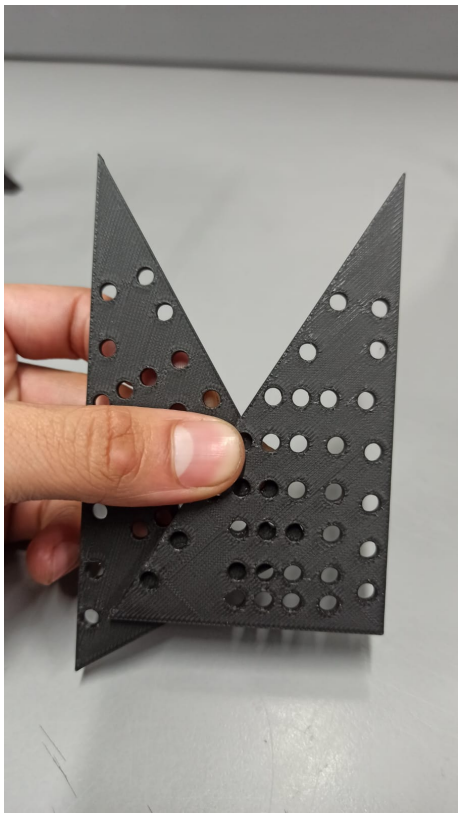

Figura 14: Paredes de la rampa

Figura 15: Tolva del prototipo

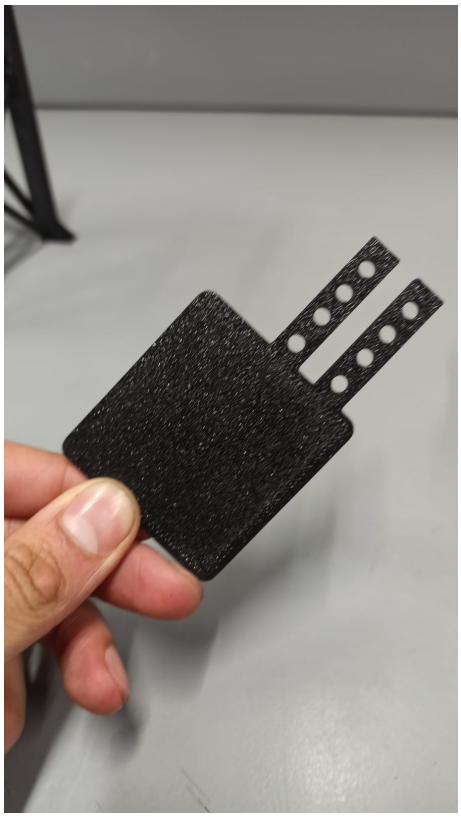

Figura 16: Trampilla movible del prototipo

Figura 17: Recipiente del prototipo

APÉNDICE B
ANEXO: CÓDIGO PS C

```c
#include "xparameters.h"
#include "xuartps.h"
#include "xil_printf.h"
#include "xgpio.h"
#include "sleep.h"
#include <string.h>

/* ---------------------- DEFINES ---------------------- */
/* UART0 (EMIO) */
#define UART0_BASEADDR   XPAR_XUARTPS_0_BASEADDR
#define UART0_CLK        XPAR_UART0_CLOCK_FREQ
#define UART_BAUDRATE    9600

/* AXI GPIO del SERVO */
#define GPIO_SERVO_BASEADDR   XPAR_XGPIO_2_BASEADDR

/* AXI GPIO del SENSOR RCWL-0516 */
#define GPIO_SENSOR_BASEADDR  XPAR_XGPIO_1_BASEADDR

/* Servo pulse (microsegundos) */
#define SERVO_PULSE_0US       1000
#define SERVO_PULSE_180US     2000
#define SERVO_HOLD_CYCLES     50

/* ------------------ Periféricos ------------------ */
XUartPs uart;
XGpio gpio_servo;
XGpio gpio_sensor;

/* ------------------ Funciones del SERVO ------------------ */

void pwm_pulse_us(unsigned int high_us) {
    XGpio_DiscreteWrite(&gpio_servo, 1, 1); // HIGH
    usleep(high_us);
    XGpio_DiscreteWrite(&gpio_servo, 1, 0); // LOW
    usleep(20000 - high_us);
}

void servo_hold_us(unsigned int pulse_us, unsigned int cycles) {
    for (unsigned int i = 0; i < cycles; i++) {
        pwm_pulse_us(pulse_us);
    }
}

void servo_to_0(void) {
    servo_hold_us(SERVO_PULSE_0US, SERVO_HOLD_CYCLES);
}

void servo_to_180(void) {
    servo_hold_us(SERVO_PULSE_180US, SERVO_HOLD_CYCLES);
```

Figura 18: Código PT.1

```c
}

/* Acción completa */
void activar_servo_evento(void) {
    xil_printf("Activando servo 180°...\r\n");
    servo_to_180();
    usleep(200000);  // pausa opcional
    xil_printf("Regresando servo a 0°...\r\n");
    servo_to_0();
}

/* ------------------ UART ------------------ */
int uart0_init(u32 baseaddr, u32 input_clk_hz, int baud)
{
    XUartPs_Config config;
    config.BaseAddress  = baseaddr;
    config.InputClockHz = input_clk_hz;

    int st = XUartPs_CfgInitialize(&uart, &config, config.BaseAddress);
    if (st != XST_SUCCESS) return -1;

    XUartPs_SetBaudRate(&uart, baud);
    return 0;
}

/* ------------------ MAIN ------------------ */
int main(void)
{
    /* SERVO GPIO */
    if (XGpio_Initialize(&gpio_servo, GPIO_SERVO_BASEADDR) != XST_SUCCESS) {
        xil_printf("Error inicializando GPIO del servo\r\n");
        return -1;
    }
    XGpio_SetDataDirection(&gpio_servo, 1, 0x0);

    /* SENSOR GPIO */
    if (XGpio_Initialize(&gpio_sensor, GPIO_SENSOR_BASEADDR) != XST_SUCCESS) {
        xil_printf("Error inicializando GPIO del sensor\r\n");
        return -1;
    }
    XGpio_SetDataDirection(&gpio_sensor, 1, 0xFFFFFFFF); // entrada

    /* UART0 */
    if (uart0_init(UART0_BASEADDR, UART0_CLK, UART_BAUDRATE) != 0) {
        xil_printf("Error inicializando UART0\r\n");
        return -1;
    }

    xil_printf("Sistema listo. Servo a 0°...\r\n");
    servo_to_0();
```

Figura 19: Código PT.2

```c
    /* ----------- CONTROL DE TIEMPOS ----------- */
    int contador_uart = 0;     // cada 1ms
    int contador_sensor = 0;   // cada 5000ms (5s)

    /* ----------- LOOP PRINCIPAL ----------- */
    while (1) {

        /* ---------------- UART cada 1ms ---------------- */
        if (contador_uart >= 1) {
            contador_uart = 0;

            if (XUartPs_IsReceiveData(UART0_BASEADDR)) {
                char c = XUartPs_RecvByte(UART0_BASEADDR);

                if (c == '1') {
                    xil_printf("[UART] Recibido '1'. Activando servo...\r\n");
                    activar_servo_evento();
                }
            }
        }

        /* ---------------- SENSOR cada 5s ---------------- */
        if (contador_sensor >= 5000) {
            contador_sensor = 0;

            u32 sensor_value = XGpio_DiscreteRead(&gpio_sensor, 1);

            if (sensor_value == 1) {
                xil_printf("[SENSOR] Movimiento detectado!\r\n");
                activar_servo_evento();
            }
        }

        /* ---- Delay base: 1ms para marcar el tiempo ---- */
        usleep(1000);  // 1 ms

        contador_uart++;
        contador_sensor++;
    }

    return 0;
}
```

Figura 20: Codigo PT.3