FPGA Based Bluetooth Controlled Land Vehicle

Ratnavalli Emani
SR No: 22548, MTech ESE
ratnavallie@iisc.ac.in

Chaithra Sri Gorrepati
SR No: 22551, MTech ESE
chaithrag@iisc.ac.in

Sai Charan Katakam
SR No: 22845, MTech ESE
saicharank@iisc.ac.in

Sri Sai Nomula
SR No: 22986, MTech ESE
srisainomula@iisc.ac.in ,

I Introduction

- This project involves employing an FPGA chip to develop a Bluetooth-controlled land vehicle. The vehicle, programmed using Verilog, is implemented on a Basys-3 FPGA board using Xilinx Vivado.
- Android offers various connectivity options including Wi-Fi, Bluetooth, and cellular data. Remote navigation and control of the vehicle are achieved via an Android smartphone utilizing Bluetooth module.
- To control the vehicle, users must first install an Android application from the Play Store onto a compatible mobile phone or device and establish a connection. The Bluetooth module, facilitated by the Host Controller Interface (HCI), enables the transmission and reception of data between the host system and the vehicle. Various commands, such as move forward/backward, start/stop, and turn left/right, are utilized for vehicle control.
- Users send these commands via the Android application to the Bluetooth receiver on the vehicle, which incorporates the FPGA board. The Android application is designed to send serial data to the Bluetooth module upon pressing any button on the application interface. At the receiving end, the vehicle's Bluetooth module forwards the data to the FPGA through the TX pin (connected to the FPGA's RX pin). The FPGA processes the navigation code received and manages the vehicle's driving process accordingly.
- Bluetooth, being an open standard specification for short-range wireless communication, enables seamless control of the vehicle. The maximum baud rate for wireless control of this Bluetooth-controlled land vehicle is 54600 bps. Here we use 9600 bps as the data speed rate.
- Based on the Received data from Bluetooth we give commands to the Motor Drive to move the vehicle in the desired direction.
- Ultrasonic Sensors are used to detect the obstacles. If there is an obstacle we need to adjust the direction of the vehicle to avoid the obstacle.

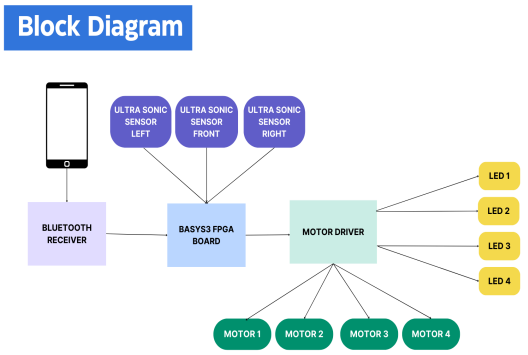

Figure 1: Flow Diagram of Bluetooth Controlled Vehicle

II Motivation

- Engaging in this project offers a hands-on learning experience in various fields such as digital design, and mobile application interfacing and has the chance to deepen the understanding of FPGA programming, Verilog, Bluetooth communication protocols.
- Tackling the challenges involved in integrating different technologies like FPGA, Bluetooth, and Android will sharpen problem-solving abilities.
- Emphasize the practical application in the form of a Bluetooth-controlled land vehicle. This can example illustrates how technology can be utilized to solve real-world problems or enhance user experiences.
- Cross-disciplinary Collaboration : combining hardware design (FPGA) with software development (Android application).This demonstrates your ability to work across different domains and integrate diverse technologies.
- This innovation limits impedance and transmission blunders and gives transmission security.

III Background Study

1. **Bluetooth Technology:** The Bluetooth-controlled interface facilitates communication between the Android smartphone and the FPGA-based vehicle control system, allowing users to send navigation commands and receive feedback on the vehicle's status.

The interface must be designed to handle the transmission and reception of data packets efficiently, ensuring reliable communication between the user and the vehicle control system.

2. **Bluetooth Modules:** Considering factors such as range, power consumption, and compatibility with the FPGA and Android smartphone and available drivers or libraries for interfacing the module with the FPGA and compatibility with the Android operating system for smartphone communication. Need to choose suitable bluetooth module.

Figure 2: HC-05 Bluetooth Module

3. **Baud Rate and Data Transmission:** Bluetooth allows the communication by the UART protocol using between different baud rates changing from 1200 to 115200 bauds. Choosing an appropriate baud rate is crucial for ensuring efficient and responsive communication between the Bluetooth module and the FPGA-based vehicle control system, as it directly impacts data transfer speed and system performance.

4. **UART Protocol:** Familiarizing with the UART protocol, which is a commonly used serial communication protocol for asynchronous communication between two devices. Understanding the structure of UART frames, including start and stop bits, data bits, and optional parity bits.

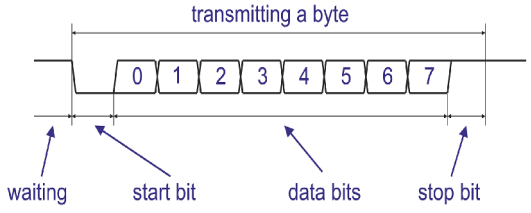

Figure 3: UART Protocol Frame Format

5. **Vehicle Control Algorithms:** Obstacle avoidance algorithms focus on detecting and maneuvering around obstacles in the vehicle's path, employing sensors such as ultrasonic sensors.

6. **Ultrasonic Sensors:** Ultrasonic sensors work based on the principle of echolocation, similar to how bats navigate in the dark. They emit high-frequency sound waves (ultrasonic pulses) and then listen for the echoes reflected off nearby objects. By measuring the time it takes for the sound waves to travel to the object and back, the sensor can calculate the distance to the object.

Figure 4: HC-SR04 Ultrasonic Sensor

7. **LM293D Motor Driver:** The LM293 is designed to control the direction and speed of DC motors. It contains two independent H-bridge driver circuits, each capable of driving one DC motor bidirectionally.

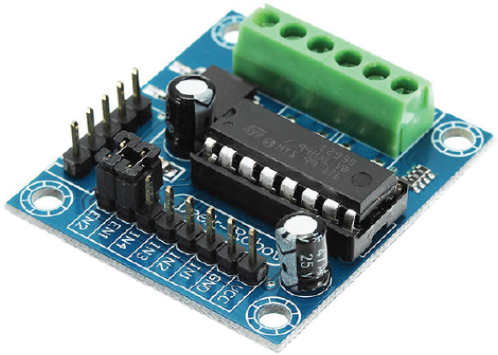

Figure 5: LM293D Motor Driver

8. **KY-006 Buzzer:** The KY-006 Buzzer module is essentially an electromagnetic buzzer that converts electrical signals into audible sound waves. It contains a piezoelectric element or a magnetic coil that vibrates when an electrical current is applied, generating sound waves in the audible frequency range.The operating voltage of the KY-006 Buzzer module typically ranges from 3V to 5V.

Figure 6: Buzzer Module

9. **Host Controller Interface (HCI):** The Host Controller Interface (HCI) serves as a crucial link between the Bluetooth module and the host system, facilitating seamless communication. It defines a set of commands and protocols used for data transmission between the host, typically the FPGA in this context, and the Bluetooth module.

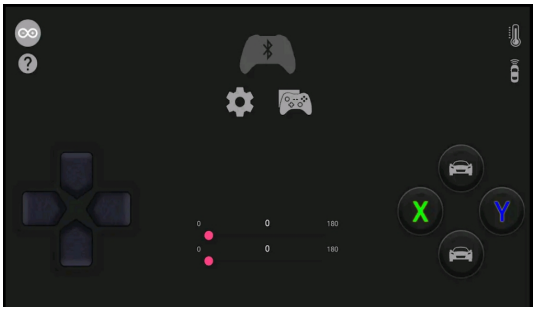

Figure 7: Mobile Application to send Commands to Bluetooth

10. **28BYJ-48 stepper motor:** The 28BYJ-48 stepper motor often comes with a built-in gearbox, usually a reduction ratio of 1:64. This means that for every 64 steps the motor takes, the output shaft rotates by one full revolution.

Figure 8: Stepper Motor to control Obstacle Picking

IV Design Space Exploration and Strategy

4.1 UART Coding:-

- Asynchronous Receiver doesn't have access to the clock [**Clock Gating**]. To determine the incoming data we look for the start bit. Once the Start bit is detected then we sample at the known baud rate to acquire the data bits.
- Counting at a higher rate (instead of 9600 bps) is going to increase speed but the numbers get bigger and the **Power Consumption** increases. So, to maintain the balance between speed of transmission and power consumption we used 9600 bps as data rate.

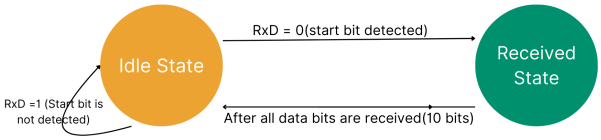

Figure 9: State Machine of UART Receiving

4.2 UART Receiver Logic:-

- If **Reset is Asserted** we need to clear all the counters such as bit counter (used to count the 10 bits [start + 8 data + stop]), counter (used to count the defined baud rate [9600]) and sample counter (used to define the over sampling of the signal).
- If **Reset is not Asserted** we will keep increasing the counter value and once we reach the defined baud rate value corresponding with clk frequency and sampling rate i.e., (clk freq/ baud rate * sampling rate) we will check for the Start bit.
- If **Start bit is detected** then we will start storing the data into the rxshiftreg by shifting the data to right once every new data bit is detected and we will increment the bit counter.
- If **Start bit is not detected** we need to continue the process by clearing all the registers.
- The registers are controlled based on the mealy state diagram shown in Fig.9

4.3 UltraSonic Sensors:-

- Ultrasonic sensors calculate distance using the time-of-flight principle.The sensor emits an ultrasonic pulse, usually at a frequency around 40 kHz. This pulse (trigger) travels through the air until it encounters an object, and When the pulse hits an object, it gets reflected back towards the sensor (echo).

- By measuring the time it takes for the pulse to travel from the sensor to the object and back, the sensor can determine the time of flight (TOF) of the ultrasonic wave.

$$Distance = 0.5 * Speed of Sound * TOF \quad (1)$$

- When an ultrasonic sensor detects an object within a specific range, it recognizes it as an obstacle and adjusts its orientation or movement accordingly.

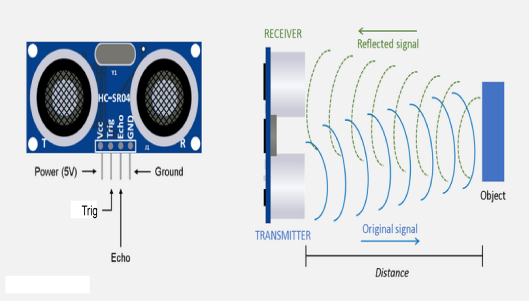

Figure 10: Working of Ultrasonic Sensor

- We provide clk to Left and Right Ultrasonic Sensors only when there is obstacle detected on the Front. This way we have done the **Clock Gating** to the Ultrasonic Sensors.

4.4 Motors Controlling:-

- Based on the Received Data from UART and Ultrasonic Sensors we drive the motor driver to control the Motors direction.
- Whenever we give some command such as forward/backward, left/right from the Android App, it gets transmitted to HC-05 Bluetooth Module and in turn to the Basys 3 Board using UART Logic. Once we receive Forward command we drive the motor driver such that bot moves forward.
- When bot is moving forward and there is an obstacle detected on the front then we immediately check the left ultrasonic sensor, If there is no obstacle on the left side we move the bot to left. But if there is obstacle on the left also then we check right ultrasonic sensor and move to right if there is no obstacle. If there is obstacle on 3 sides then we stop the bot.
- We have also designed a obstacle picking mechanism at the front of the bot to pick smaller obstacles, so that bot can continue moving.

4.5 Programming Basys3 Board's Non-Volatile Flash Memory:-

- When the BASYS3 board ships it comes with a diagnostic program stored in its SPI Flash Memory. Once the board is turned on, by for example powering it through its USB port, this program is loaded into the FPGA. The BASYS3's HEX display lights up with flashing numbers and the LEDs turn on when the corresponding switches are turned on and other.
- The BASYS3 boards can then be programmed through bitstream files generated by Xilinx's Vivado from either an HDL language, such as Verilog or VHDL, or from a Block Design. Vivado's built in Hardware Manager provides the means to program the boards through its USB-JTAG circuitry.
- By default, the bitstream is sent through the USB cable to the (volatile) SRAM-based memory cells within the FPGA where it remains until a) it is overwritten by a new bitstream, b) the board is reset, or, c) turned off. In cases b) and c), the board will revert to the bitstream stored in its SPI Flash memory.
- However, it is possible to change the default settings to program the non-volatile SPI Flash memory using Vivado's Hardware manager. In that case, when the board is powered up (or reset) it will always start with the last bitstream stored in its Flash memory. This is extremely useful as our project is a standalone application on the BASYS3 board which need to start on Power-ON without the aid of a PC.
- So, we have accessed the Non-Volatile Flash Memory of Basys3 Board and erased the default code provided by the manufacturer and programmed with out Verilog Code.

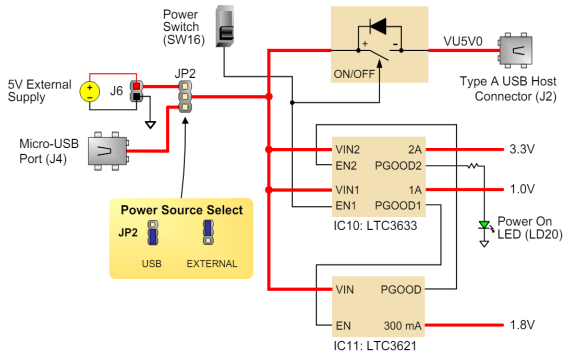

Figure 11: Basys3 Power Circuit

4.6 External Power Supply to the Basys3 Board:-

- As we wanted our bot to run without any connection from PC or any long cable so that we dont want to move along with the bot carrying the cable connected to Basys 3 Board we have provided the power supply to board externally using battery.

- An external power supply can be used by plugging into the external power header (J6) and setting jumper JP2 to "EXT". The supply must deliver 4.5VDC to 5.5VDC and at least 1A of current (i.e., at least 5W of power).
- The power provided to USB devices that are connected to Host connector J2 is not regulated. Therefore it is necessary to limit the maximum voltage of an external battery pack to 5.5V DC.

V Implementation Challenges

1. **Variations in Bluetooth Module:-** There are 2 types of HC-05 with some different capabilities. One HC-05 Works fine and performs the desired operation whereas, another HC-05 is unable to communicate. Identified these differences using Arduino. Module shown in Fig-(12) is able to communicate and able to drive whereas Fig-(13) is unable to communicate.

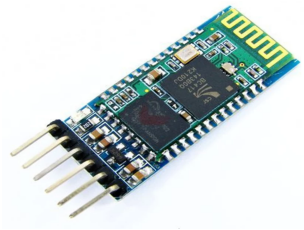

Figure 12: HC-05 Bluetooth Module

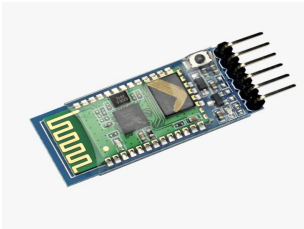

Figure 13: HC-05 Bluetooth Module

2. **Driving Motor Driver:-** FPGA is unable to drive motor driver(L298) and is able to drive motor driver (L293D).

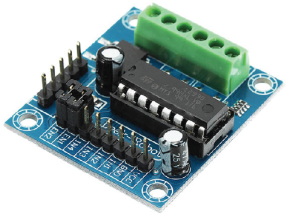

Figure 14: L293D Motor Driver

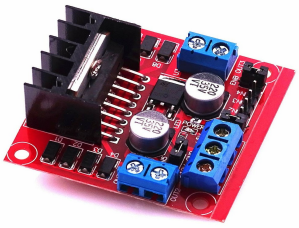

Figure 15: L298 Motor Driver

3. **Reliability of Jumper Wires:-**Jumper Wires were getting disconnected (Not shorted internally) frequently. Making us to rewire the entire circuit again and again.

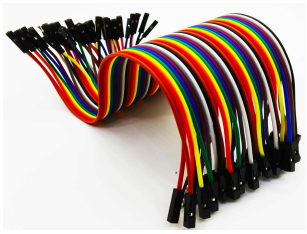

Figure 16: Jumper Wires

4. **Unable to drive Ultrasonic Sensors** which need 5V Supply, from FPGA Board we can provide only 3.3V. Thus, needed to test Ultrasonic sensors with Arduino initially, which made us to code for the Arduino also. As ultrasonic sensors do not have any indication of light, to check whether they are working or not.
5. As we can't test our motors every single time directly, we had to first check it with FPGA LED's and modify code accordingly for motors.
6. **For Bluetooth interfacing**, we should use **specific pins from the Basys3 Board**. We were unaware of it and tried many ways to connect it, which made us to go through datasheets and digilent blog for proper mapping.
7. Placement of many hardware components on the chassis of our vehicle was challenging with multiple (almost every) port being used from Basys3 Board.
8. To interface the Obstacle picking component with FPGA, **making the picking hardware setup** was so hard to make which included mechanical stuff to give it a shape.
9. Throughout the development process, various challenges were encountered and overcome, including debugging communication protocols, optimizing sensor integration, and refining obstacle avoidance algorithms. These challenges provided valuable learning

opportunities and insights into the complexities of interdisciplinary projects involving hardware-software integration and real-time control systems.

VI RESULTS

6.1 *RTL Blocks:-*

The RTL component statistics and schematics of the Bluetooth Controlled Land Vehicle designed using Verilog Code gets synthesized into the Hardware as shown below in Fig.17.

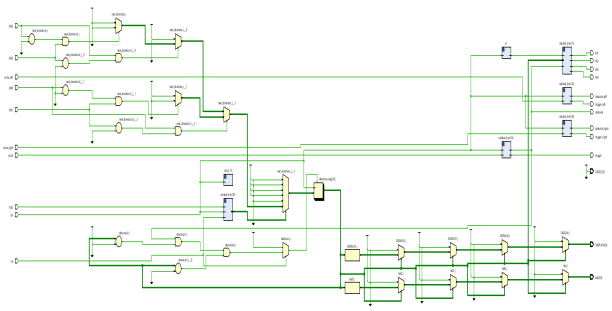

Figure 17: RTL Block of Bluetooth Controlled Vehicle

Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	Bonded IOB (106)	BUFGCTRL (32)
N main	1029	553	504	1029	35	1
a1 (oneHZ_generator)	8	27	10	8	0	0
nolabel_line103 (bluetooth_RX)	22	34	13	22	0	0
nolabel_line113 (motor_control)	7	8	3	7	0	0
nolabel_line134 (us_sensor)	330	161	160	330	0	0
nolabel_line135 (us_sensor_0)	330	160	160	330	0	0
nolabel_line136 (us_sensor_1)	330	160	158	330	0	0

Figure 18: Resource Utilization

6.2 *Timing Report:-*

Total Negative Slack of Setup time and Total Hold Slack are 0ns each. This implies there are NO timing violations.These reports offer crucial insights into the behavior and efficiency of the synthesized circuit within the target hardware environment.

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.105 ns	Worst Hold Slack (WHS):	0.055 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	787	Total Number of Endpoints:	787	Total Number of Endpoints:	542

All user specified timing constraints are met.

Figure 19: Timing Summary

6.3 *Bluetooth Controlled Vehicle:-*

The designed Vehicle using the chassis, tyres, ultrasonic sensors, buzzer, LEDs, obstacle picker , Power Bank for the Supply and a lot of wires.

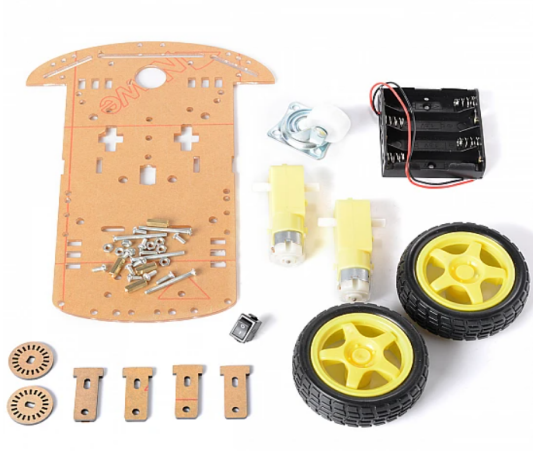

Figure 20: Raw Chassis Frame for the vehicle

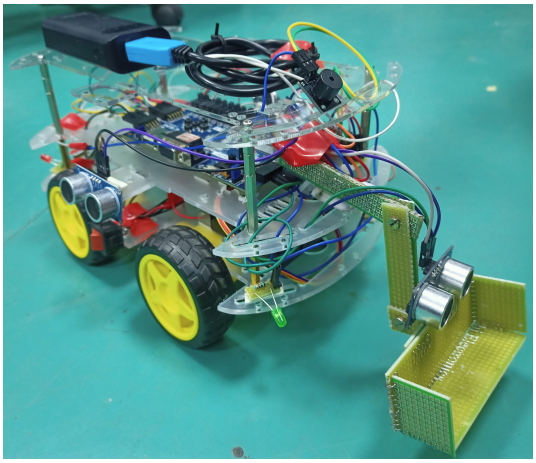

Figure 21: Final Bluetooth Controlled Vehicle

VII FUTURE SCOPE

- **Gesture-Based Control:-** Explore the integration of gesture recognition technology for controlling the vehicle's movement. By analyzing hand gestures captured by the smartphone's camera, users could intuitively command the vehicle to move in different directions or perform specific actions.
- **Multi-Vehicle Coordination:-** Investigate the feasibility of coordinating multiple FPGA-based vehicles to work together on collaborative tasks. Implementing communication protocols and coordination algorithms would enable teams of vehicles to collaborate on tasks such as exploration, surveillance, or distributed sensing.
- **Energy Efficiency Optimization:-** Optimize the vehicle's energy consumption through efficient power management techniques and the integration of energy harvesting technologies. This could include implementing sleep modes, regenerative braking systems, or solar panels to extend the vehicle's operating time

and reduce reliance on external power sources.

- **Integration of GPS:-** Incorporate GPS functionality into the system to enable navigation and tracking capabilities. This would allow the vehicle to autonomously navigate predefined routes or follow GPS coordinates, expanding its range of applications for outdoor environments.
- **Enhanced Obstacle Avoidance Algorithms:-** Explore and implement advanced obstacle avoidance algorithms using machine learning or computer vision techniques. This could involve integrating additional sensors such as cameras or LiDAR for environment perception, allowing the vehicle to make more informed decisions in navigating complex surroundings.

VIII Conclusion

Through the integration of hardware and software components, this project successfully demonstrates the seamless communication and coordination between different technologies to enable remote navigation and obstacle avoidance. We have Successfully integrated Bluetooth communication between an Android smartphone and the FPGA-based vehicle, enabling wireless remote control. And then Incorporation of ultrasonic sensors for obstacle detection, allowing the vehicle to autonomously navigate its environment and change direction to avoid collisions.We have Activated buzzer as a real-time alert mechanism to notify users when an obstacle is detected, enhancing safety and situational awareness during operation. We have also developed voice command capabilities for controlling the vehicle using voice recognition technology. This would allow users to interact with the vehicle hands-free, issuing commands verbally through the Android app. By leveraging the power of FPGA-based hardware acceleration and wireless communication, this project opens up new possibilities for remote control, autonomous navigation, and obstacle avoidance in robotics and beyond.

References

[1] R. Z. Eshita, T. Barua, A. Barua, A. M. Dip,"Bluetooth Based Android Controlled Robot," American Journal of Engineering Research, vol. 5, pp. 195ˆ a199, 2016.

[2] Ameen Majeed, Salih Baris Ozturk, and Didem Kivanc Tureli, "An Encoder Fault Tolerant FPGA Based Robot Control Using Bluetooth of a Smart Phone", IEEE Conferences, pp. 1336ˆ a1341, 2017

[3] WW. Kong, "uart- universial asynchronous receiver and transmitter", RH2T Magazine Vol.3, Dec 2010.

[4] "Interfacing Ultrasonic Sensor with FPGA"

[5] A. Giorgio, F. Paris, "Design of a reliable Bluetooth interface for FPGA-based Embedded Systems," International Journal of Advanced Engineering Research and Applications (IJA-ERA), Volume ˆ a 3, Issue ˆ a 5, September ˆ a 2017.

[6] A. M. Gibb, New media art, design, and the Arduino microcontroller: A malleable tool. PhD thesis, Pratt Institute, 2010. [9] Ameen Majeed, Salih Baris Ozturk, and Didem Kivanc Tureli, "An Encoder Fault Tolerant FPGA Based Robot Control Using Bluetooth of a Smart Phone", IEEE Conferences, pp. 1336ˆ a1341, 201

[7] "Basys 3 Reference Manual"

[8] "Digilent Blog"