LAMBTON COLLEGE

Embedded Systems and Engineering Design

PROJECT REPORT

PIXY BOT ROBOT

A 4th Semester Project for post-graduate Diploma for

Embedded Systems and Engineering Design

By:

Jaspreet Singh C0730778

Ankit Jhall C0727969

Submitted To:

Takis Zourntos

ABSTRACT

Pixy tracking robot is an autonomous robot which is capable of recognizing object shape, size, and color. It's trained for specific objects, and then it can follow them. It is designed for automatic behavior and follow objects. The robot use Pixy2 cam for the vision it is a really powerful vision system it can learn objects which you teach. The robot is driven by DC motors controlled by LPC54114 from NXP. All the algorithms take place on LPC54114, which is responsible for the control of motors and turning the robot. The Pixy2 is controlled via Pocket Beagle a tiny computer with powerful specification running Linux. So, it can be said that this robot has two different brains interconnected with each other. One brain is controlling the vision of a robot while other controls the movement. This project aim is to make a robot which can follow specific objects as well as avoiding any obstacles. The robot has different behaviors like if no trained object is insight the robot executes wandering behavior to search objects and randomly move in any direction for searching.

ACKNOWLEDGMENT

We want to express our thanks to, Prof Takis Zourntos, for going through every detail, giving expert advice and encouragement throughout this difficult project to make this project a success.

Last but not least, we would like to thank our friends and family for their help in every way for the success of this project.

ABBREVIATIONS

FPS – Frames per second.

MCU – Microcontrollers

PRUS – Programmable Real Time Units

DC – Direct Current

PWM – Pulse Width Modulation

SPI – Serial Peripheral Interface

UART – Universal Asynchronous Receiver/Transmitter

RAM- Random Access Memory

1. Introduction

Pixy tracking robot is a machine which can follow objects like balls etc. It is a really powerful robot consisting of two MCU interconnected with each other, i.e., Pocket beagle and LPC54114. The robot continuously tracks the object with Pixy2 and send information to LPC via Pocket beagle. The key components here is Pixy2, LPC54114, and Pocket Beagle. The Pixy extracts the data from the environment and sent it to Pocket Beagle which is interconnected with LPC54114 after getting the data from Pocket beagle, LPC54114 activates the DC motors so that robot can move and follow the object.

2. Hardware Description

It can be said that hardware is divided into two parts high level and low level. Pixy2, LPC54114 and Pocket beagle are in the category of high-level hardware rest of the hardware is at a low level.

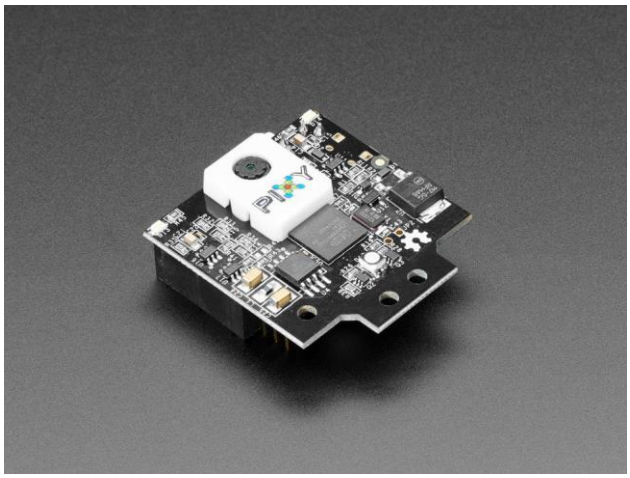
Components used in this project are the following:

1. LPC54114 by NXP
2. Pocket Beagle
3. Pixy 2
4. Zumo Chassis
5. DC motors
6. Ultrasonic Sensor
7. Lithium-Ion Batteries
8. Adjustable Step-Up Regulator
9. HM-10 Bluetooth
10. DRV 8833 Motor Driver

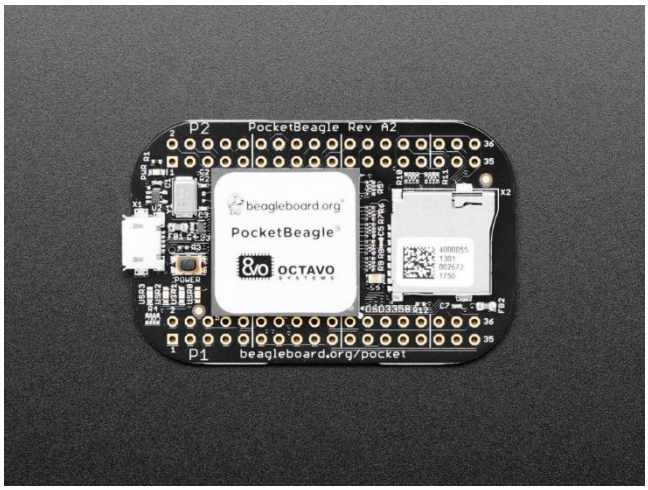
LPC54114: It is a very powerful and power-efficient MCU. Based on power-efficient Cortex-M4 core, with an optional M0 coprocessor. It runs a FreeRTOS application and fully supports

MCUXpresso software

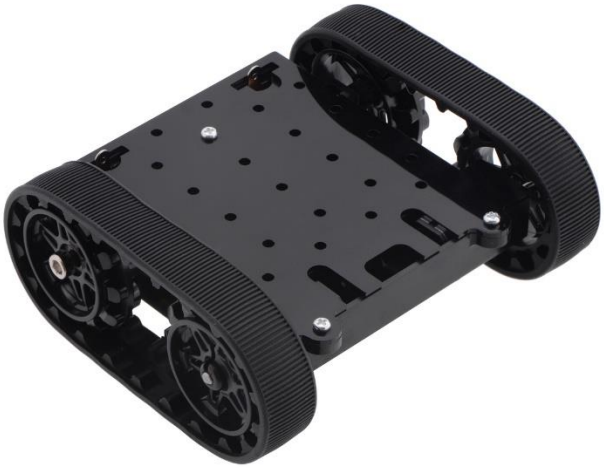
PIXY2: It is a very small, faster, and reliable vision system. It can learn and detect objects which we teach. It has new algorithms which can detect intersections, road signs, etc. It is capable of doing this at 60fps. It can be taught seven different signatures.



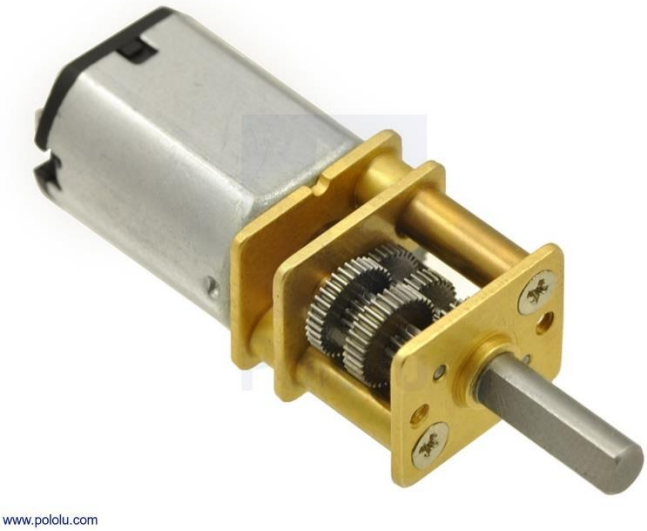
POCKET BEAGLE: It is an ultra-tiny computer with very powerful specification running Linux. 1ghz processor Cortex-A8, 512 MB RAM, 2x PRU'S.



Zumo Chassis: It is used for placing all the components together on it. It features a compartment for 4AA batteries slot and a differential drive system with no motors.

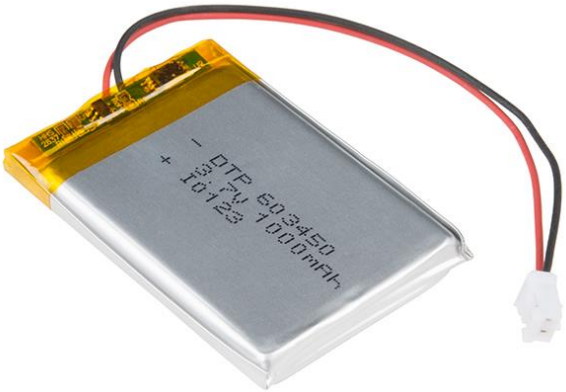


DC Motors: Two DC motors 75:1 micro metal gearmotors are used in this project running at 6V.

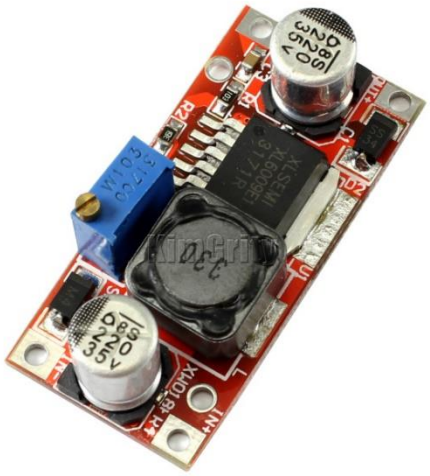


Ultrasonic Sensor: It is used for avoiding obstacles the operating range is between 2-500 cm with a 30-degree angle. It uses sonar to determine the distance of the object. Some animals, like bats, do the same thing.

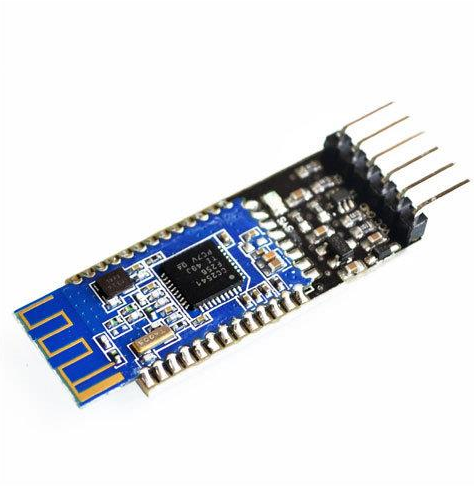
Lithium-Ion Batteries: Two lithium batteries used 2500mah and 2000mah total of 4500mah. It is rechargeable.



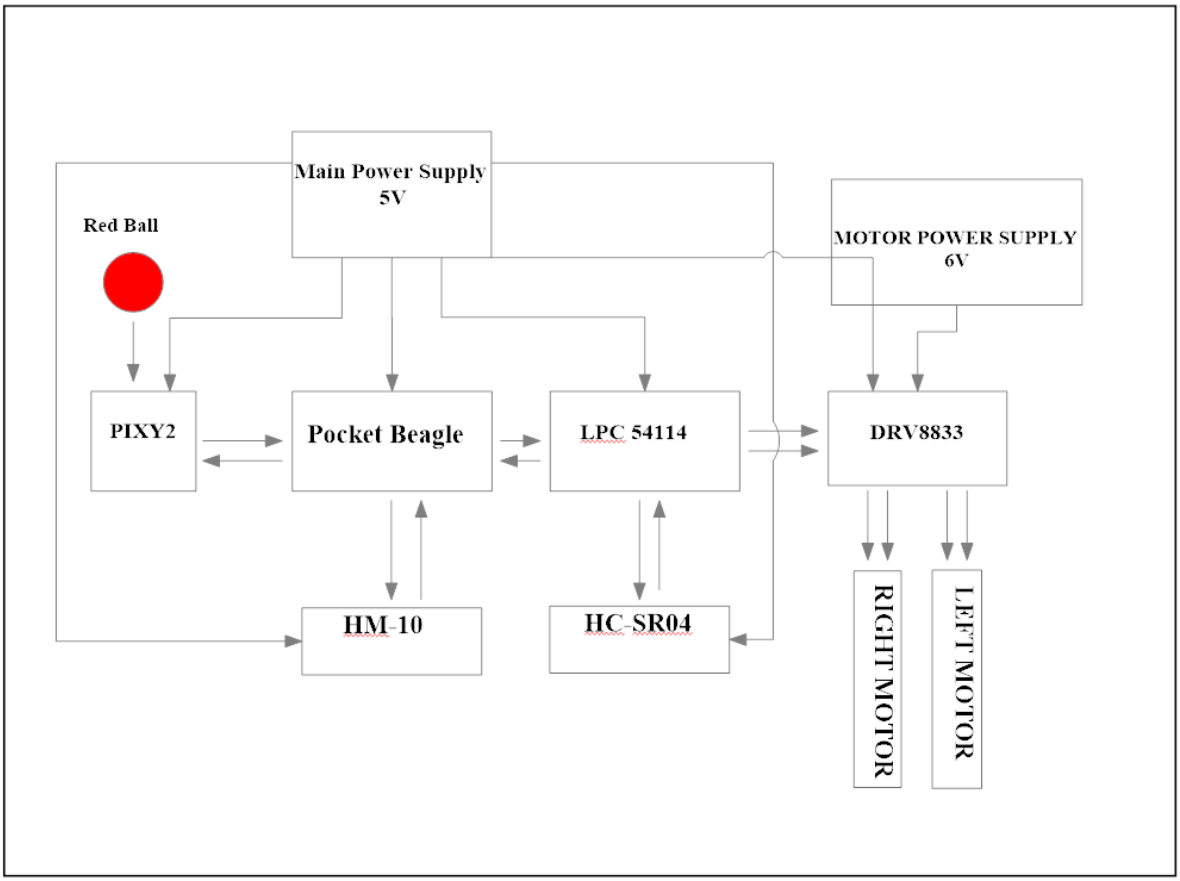
Adjustable step-up regulator: Total two of these are used in the project. One is used for the DC motors at 6V other one is used to supply 5V to all the other components like LPC54114, Pocket Beagle, Pixy, sensors, etc. It can provide output from 4-35V with load up to 3A.



HM-10 & DRV8833: Bluetooth is used to communicate with the robot via Smartphone, and DRV8833 is a motor driver which is used in between the pins of



BASIC BLOCK DIAGRAM:

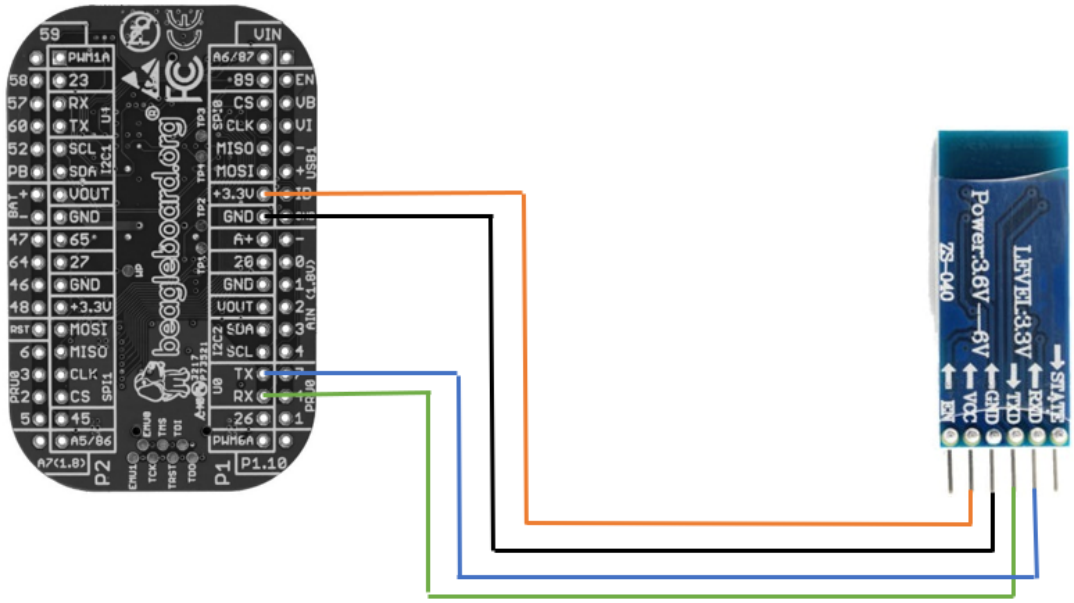


Working in Steps:

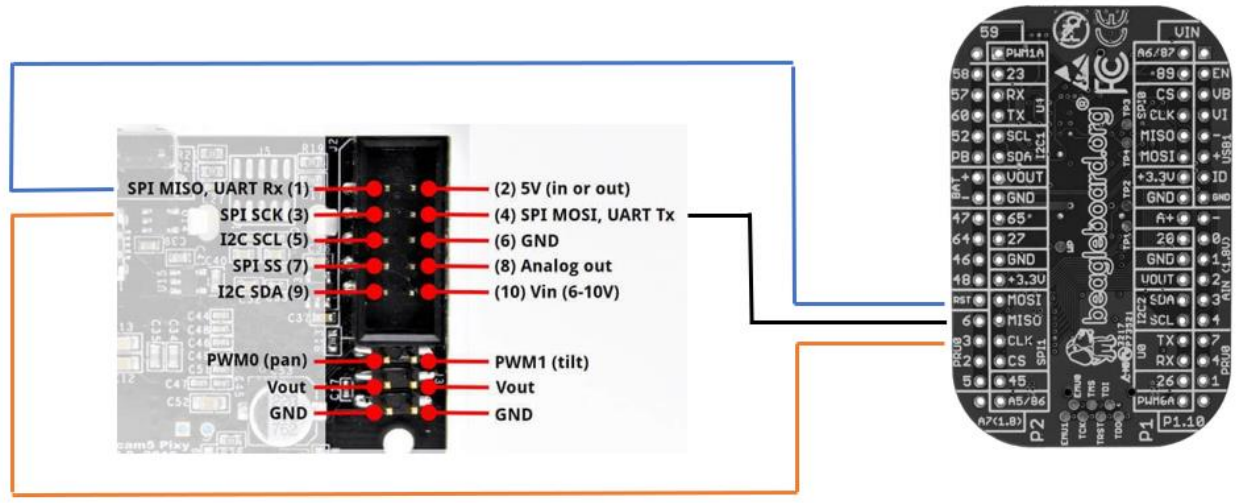
1. Pixy detects the Red ball convert its data into text form and send it to Pocket Beagle.
2. Pocket Beagle is connected to LPC54114 via UART; it sends the text information from pixy to LPC54114.
3. After receiving the information from Pocket Beagle, LPC54114 send PWM pulses to the DRV8833.
4. DRV8833 turns ON the DC motors.

Here HM-10 Bluetooth is used if we want to control the robot via Bluetooth and HC-SR04 (ultrasonic sensors) are used to avoid obstacles. 6V is used for DC motor of the system is operating on 5V.

Bluetooth connection with HM-10: Here Orange wire is used for VCC, Black for GND, TX of Pocket Beagle is connected to RX of Bluetooth, RX of Pocket Beagle is connected to TX of Bluetooth.



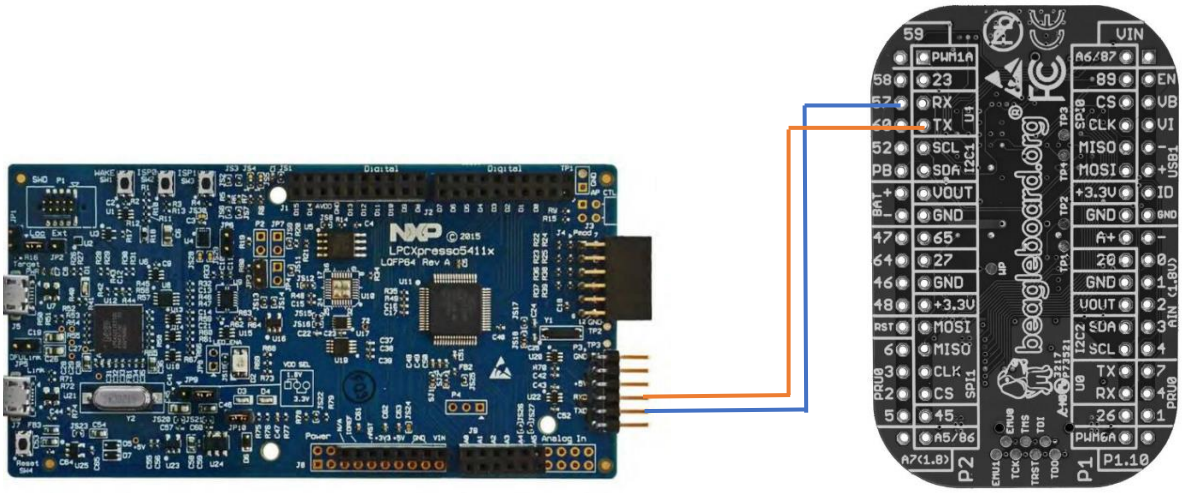
Pixy2 connection with Pocket Beagle: Spi is used for connection of Pocket Beagle and Pixy2.



Pocket Beagle and LPC54114 Interfacing:

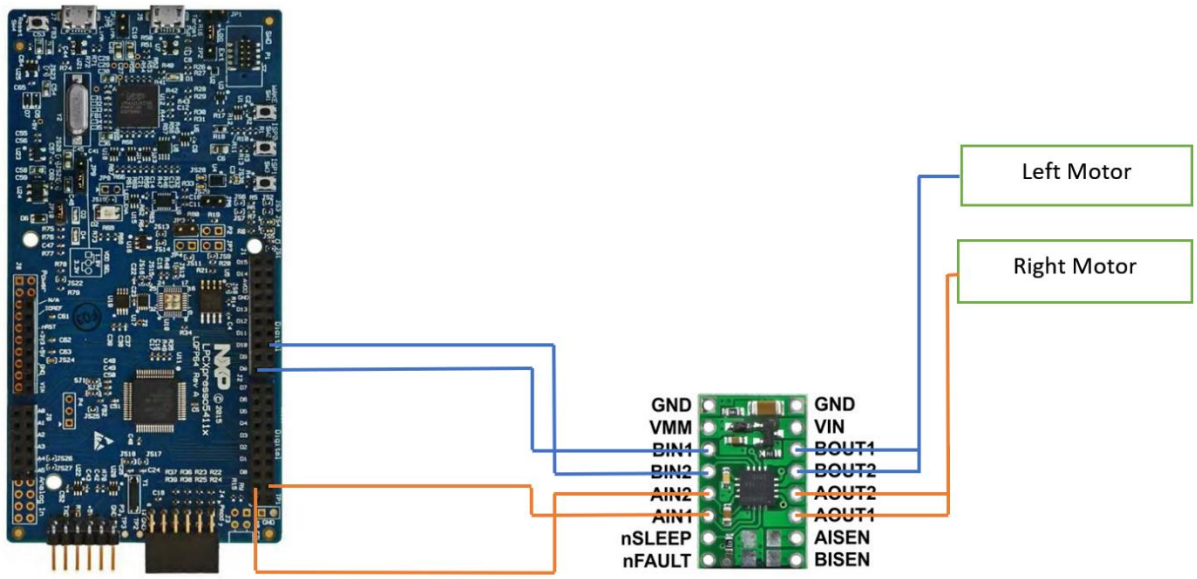
1. Both of the MCU's are really powerful with their advantages
2. Pocket beagle is clocked at 1ghz with dedicated 512MB RAM.
3. LPC54114 is very power-efficient MCU clocked up to 100mhz.
4. UART communication is used for interfacing.
5. LPC54114 runs freeRTOS application
6. Beagle Bone operated on LINUX, programming done in C++.

The orange and blue line indicates the TX and RX.

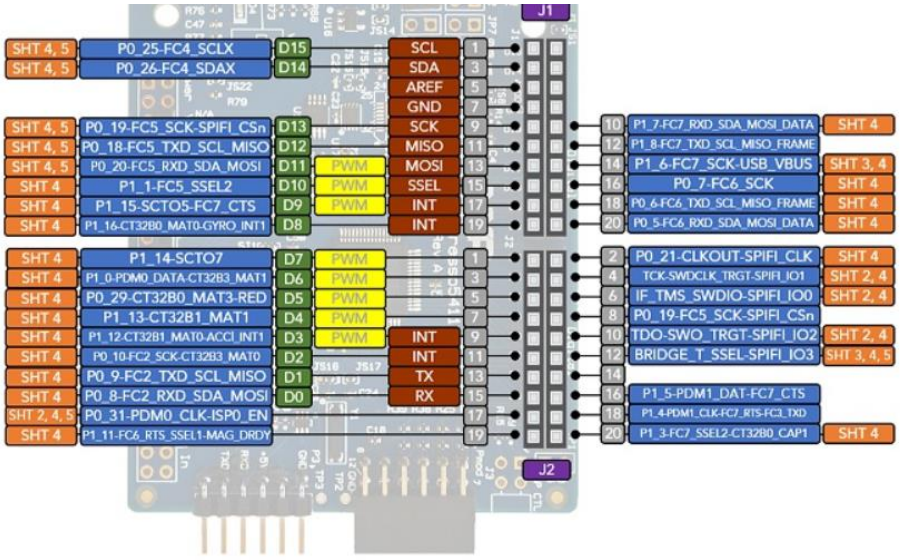


LPC54114 Connection with DRV8833: LPC sends the PWM signal to the motor driver which is responsible for the movement of motors. The specific PWM pins used are below

1. J1 PIN 19 is connected to BIN1.
2. J1 PIN 16 is connected to BIN2.
3. J2 PIN 18 is connected to AIN1.
4. J2 PIN 17 is connected to AIN2.



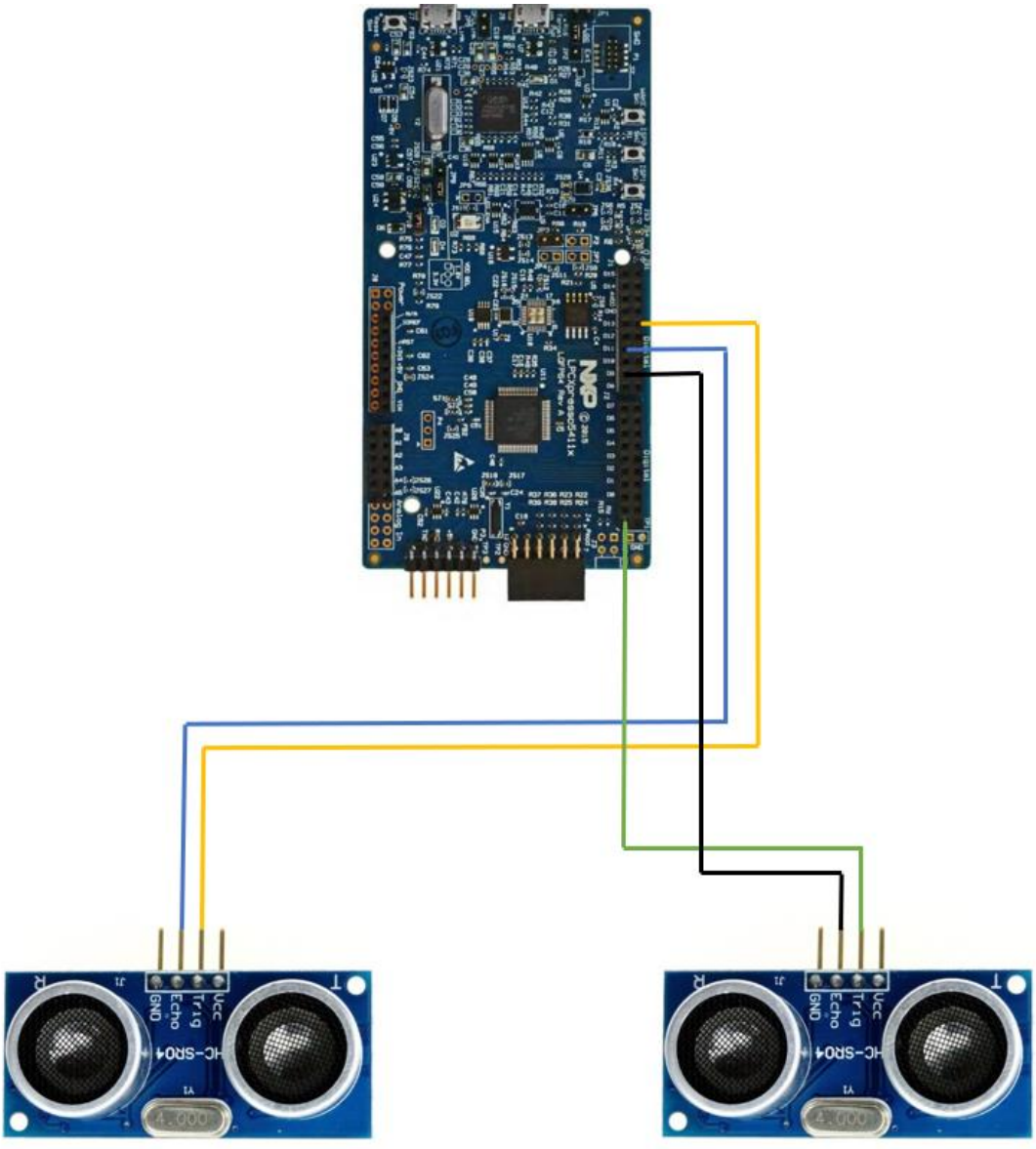
PINs of J1&J2 are:



LPC54114 Connection with Ultrasonic sensors: Two Ultrasonic sensors are used for Obstacle avoidance one in front one for the rear. The connection shown is only for the trigger pins and echo pings. GND and VCC not shown. The pins used for ultrasonic sensors are

J1 PIN 10 for Trigger pin of Sensor 1, J1 PIN 13 for Echo pin of Sensor 1.

J2 PIN 19 for Trigger pin of Sensor 2, J1 PIN 17 for Echo pin of Sensor 2.



1 **2**

Software Description:

The most complicated thing about this project is software, i.e. coding. We had two MCUs, running different platform. For example, Pocket beagle is on LINUX whereas LPC54114 is on FreeRTOS. The language used in this project is C, C++, and Embedded C. The high-level code is responsible for the movement of robot and vision system. Collecting data from vision system and turning the motors. So, two different high-level code is used while the low-level code is used for controlling other peripherals like Bluetooth, Ultrasonic sensors, etc.

The following software was used for the development of code

1. MCUXpresso IDE
2. Eclipse

We developed a FreeRTOS application running on LPC54114, coded by using Embedded C. It is responsible for controlling the DC motors, Ultrasonic sensors both of the tasks which are very important for the robot.

1. The code consists of a total of four tasks controlled by scheduler.
2. Ultrasonic task
3. UART task
4. Drive task
5. Object search
6. Each task has different priorities.

This code is just screenshots of the code. For the CPP file or PDF file, please visit GitHub links:

Jass Khaira: https://github.com/jasskhaira

Ankit Jhall: https://github.com/ankitjhall

FreeRTOS application code of LPC54114:

This code consists of 4 tasks

UART task- it is responsible for receiving data from pocket beagle via UART port and after that it sends the received data to other tasks. It also suspends and resumes other tasks according to the priorities.

Drive taks: Tt is responsible for the differential drive system of robot it receives the data from UART task via a queue and drive the motor according to the received data.

Ultrasonic task: This task is of highest priority task it is used to avoid obstacles from front and rear whenever it encounters the obstacle it activates its obstacle avoidance behavior.

Object search: It only activates when there is no signature detected by pixy. It is responsible to drive robot in such a way that it can find the desired object.

```c
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC54114_cm4.h"
#include "fsl_debug_console.h"


/* FreeRTOS kernel includes. */
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "timers.h"

#include "fsl_usart_freertos.h"
#include "fsl_usart.h"

#include "fsl_ctimer.h"

/* TODO: insert other definitions and declarations here. */

#define CTIMER CTIMER0                  /* Timer 0 */
#define LM0 kCTIMER_Match_0         //  J1[19]                  PWM Pin connected to left motor PIN1
#define RM0 kCTIMER_Match_1         //  J2[18]                  PWM Pin connected to left motor PIN2
#define LM1 kCTIMER_Match_2         //  J1[16]                  PWM Pin connected to right motor PIN1
#define RM1 kCTIMER_Match_0         //  J2[17]                  PWM Pin connected to right motor PIN2


#define DEMO_USART USART0
#define DEMO_USART_IRQHandler FLEXCOMM0_IRQHandler
#define DEMO_USART_IRQn FLEXCOMM0_IRQn


/* Task priorities. */
#define uart_task_PRIORITY (configMAX_PRIORITIES - 1)
#define USART_NVIC_PRIO 5

static void uart_task(void *pvParameters);              //Task responsible for receiving data from beaglebone
static void Drive_task(void *pvParameters);             //This Task is used to drive motor
static void Ultrasonic_Task(void *pvParameters);        //This Task associate with Ultarsonic Sensor to avoid obstracle
static void Object Search();                            //This Task used to search teh object


void MotorsSetup();                                     // It set up PWM to drive motors
void Move();                                            // It is used to move in forward direction
void Turn_SlowLeft();                                   // It is used to turn left with slow speed
void Turn_SlowRight();                                  // It is used to turn right with slow speed
void Turn_Left();                                       // It is used to turn left
```

```
52 void Turn_Right();                                    // It is used to turn right
53 void Stop();                                          // It is used to Stop robot
54 void Reverse();                                       // It is used to reverse the robot
55 float Front_Obstarcle();                              // It gives the distance of front
   obstacle
56 float Rear_Obstarcle();                               // It gives the distance of rear obstacle
57 void Search();                                        // It drive the motors to search object
58 void Circle();                                        // It drive the motors to make circle
59
60
61 uint8_t background_buffer[100];                       // For receiving data from Beaglebone in
   background
62 uint8_t recv_buffer[1];
63
64 usart_rtos_handle_t handle;                           // USART handle
65 struct _usart_handle t_handle;
66
67
68 /*This structure contain the configurations of USART
69  * USART runs at 9600 baudrate 8N1
70  */
71
72 struct rtos_usart_config usart_config = {
73     .baudrate    = 9600,
74     .parity      = kUSART_ParityDisabled,
75     .stopbits    = kUSART_OneStopBit,
76     .buffer      = background_buffer,
77     .buffer_size = sizeof(background_buffer),
78 };
79
80 /* Queue Handle  */
81
82
83 xQueueHandle Obj_track= NULL;
84 xQueueHandle queue1= NULL;
85
86 /* Task Handles   */
87
88 TaskHandle_t Uart_Task_Handle=NULL;
89 TaskHandle_t Ultrasonic_Task_Handle=NULL;
90 TaskHandle_t Drive_task_Handle=NULL;
91 TaskHandle_t Object_Search_Handle=NULL;
92
93
94 int main(void)
95
96 {
97
98
99
100     CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);
101     SYSCON->ASYNCAPBCTRL = 1;
102
103
104
105     BOARD_InitBootPins();
106     BOARD_InitBootClocks();
107     BOARD_InitBootPeripherals();
108
109
```

```
110     MotorsSetup();
111
112     /* Creating Queues for InterTask communication */
113
114     queue1=xQueueCreate(1,sizeof(uint8_t));
115     Obj_track=xQueueCreate(1,sizeof(uint8_t));
116
117
118     /* Creating Tasks for rtos */
119
120     if (xTaskCreate(uart_task, "Uart_task", configMINIMAL_STACK_SIZE + 10, NULL, 3
121             ,&Uart_Task_Handle) != pdPASS)
122         {
123             PRINTF("Task creation failed!.\r\n");
124             while (1)
125                 ;
126         }
127
128
129     if (xTaskCreate(Drive_task, "Robot_driving_task", configMINIMAL_STACK_SIZE + 10, NULL,
    2,&Drive_task_Handle) != pdPASS)
130                 {
131                     PRINTF("Task creation failed!.\r\n");
132                     while (1)
133                         ;
134                 }
135
136     if (xTaskCreate(Ultrasonic_Task, "Ultrasonic_Task", configMINIMAL_STACK_SIZE + 10, NULL,
    4,&Ultrasonic_Task_Handle) != pdPASS)
137                             {
138                                 PRINTF("Task creation failed!.\r\n");
139                                 while (1)
140                                     ;
141                             }
142     if (xTaskCreate(Object_Search, "Object_Search", configMINIMAL_STACK_SIZE + 10, NULL,
    0,&Object_Search_Handle) != pdPASS)
143                     {
144                         PRINTF("Task creation failed!.\r\n");
145                         while (1);
146
147
148                 }
149
150
151
152         vTaskStartScheduler();
153         for (;;)
154             ;
155     }
156
157
158
159
160
161 static void uart_task(void *pvParameters)
162 {
163
164     vTaskSuspend(Object_Search_Handle);
165     int error,status=0;
166     uint8_t send;
```

```
167     size_t n            = 0;
168     usart_config.srcclk = BOARD_DEBUG_UART_CLK_FREQ;
169     usart_config.base   = DEMO_USART;
170
171     NVIC_SetPriority(DEMO_USART_IRQn, USART_NVIC_PRIO);
172
173
174
175     USART_RTOS_Init(&handle, &t_handle, &usart_config);
176
177
178     while(1)
179     {
180         /* Receive the data form USART */
181
182         error=USART_RTOS_Receive(&handle, recv_buffer, sizeof(recv_buffer), &n);
183         //printf("%c",recv_buffer);
184         if (error == kStatus_USART_RxRingBufferOverrun)
185                 {
186             printf("Buffer overrun");
187
188                 }
189
190
191         if (n > 0)
192         {
193             send=recv_buffer[0];
194
195             if(send=='F')
196            {
197                 //send1=recv_buffer[0];
198                 if(status==0)
199                 {
200                 vTaskSuspend(Drive_task_Handle);
201                 vTaskResume(Object_Search_Handle);
202                 //printf("drive suspend\n");
203                 //printf("search resume\n");
204                 status=1;
205                 }
206                 xQueueSend(Obj_track,&send,10);
207
208
209            }
210             else if(send=='S')
211              {
212                 Stop();
213                vTaskSuspend(Object Search Handle);
214
215              }
216
217              else
218               {
219                  if(status==1)
220                  {
221                    vTaskSuspend(Object_Search_Handle);
222                    vTaskResume(Drive_task_Handle);
223                    //printf("drive resume\n");
224                    //printf("search suspend\n");
225                      status=0;
226
```

```
227                 }
228
229                 xQueueSend(queue1,&send,10);
230                }
231
232
233
234             //printf("%c",recv_buffer);
235         }
236     }
237
238 }
239
240
241
242 static void Drive_task(void *pvParameters)
243 {
244     uint8_t recv;
245
246        while(1){
247             xQueueReceive(queue1,&recv,10);
248             if(recv=='M')
249             {
250                 Move(90);
251                 //printf("moving\n");
252                 recv='n';
253             }
254             else if(recv=='L')
255             {
256                 Turn_Left();
257                 //printf("left\n");
258                 recv='n';
259             }
260             else if(recv=='R')
261             {
262                 Turn_Right();
263                 //printf("right\n");
264                 recv='n';
265             }
266             else if(recv=='S')
267             {
268                 Stop();
269                 //printf("stop");
270                 recv='n';
271
272             }
273             else if(recv=='B')
274             {
275                 Stop();
276                 vTaskDelay(5);
277                  Reverse();
278                 // printf("reverse\n");
279                 recv='n';
280             }
281
282             else if(recv=='l')
283             {
284                 Turn_SlowLeft();
285                 // printf("reverse\n");
286                  recv='n';
```

```
                    }
            else if(recv=='r')
            {
                Turn_SlowRight();
                // printf("reverse\n");
                 recv='n';
            }
            else if(recv=='F')
            {
                xQueueSend(Obj_track,&recv,10);
                recv='n';
            }
            }
}


static void Ultrasonic_Task(void *pvParameters)
{
   float Front_obs,Rear_obs;


    while(1)


   {
        Front_obs=Front_Obstarcle();
        Rear_obs=Rear_Obstarcle();
        if(Front_obs<10)
        {  Stop();
            vTaskSuspend(Uart_Task_Handle);
            vTaskSuspend(Drive_task_Handle);
            vTaskSuspend(Object_Search_Handle);

            vTaskDelay(10);
            Reverse();
            vTaskDelay(150);
            /*Turn_Left();
            vTaskDelay(200);
            Turn_Right();
            vTaskDelay(200);*/
            Stop();
            vTaskResume(Uart_Task_Handle);
            vTaskResume(Drive_task_Handle);
            vTaskResume(Object_Search_Handle);
           }

        if(Rear_obs<10)
        {  Stop();
            vTaskSuspend(Uart_Task_Handle);
            vTaskSuspend(Object_Search_Handle);
            vTaskResume(Drive_task_Handle);
            vTaskDelay(5);
            Move(90);
            vTaskDelay(150);
           // Turn_Left();
            //vTaskDelay(200);
            //Move();
            //vTaskDelay(80);
            //Turn_Right();
```

```
347             //vTaskDelay(100);
348             Stop();
349             vTaskResume(Uart_Task_Handle);
350             vTaskResume(Object_Search_Handle);
351             vTaskResume(Drive_task_Handle);
352            }
353
354
355
356
357
358
359    }
360
361 }
362
363
364 static void Object_Search(void *pvParameters)
365 {
366     uint8_t Obj_recv;
367    while(1)
368    {
369        printf("finding\n");
370    xQueueReceive(Obj_track,&Obj_recv,10);
371    if(Obj_recv=='F')
372    {
373    Circle();
374    Move(75);
375    vTaskDelay(300);
376    Search();
377    Stop();
378
379
380    }
381    Obj_recv='n';
382 }}
383
384
385
386
387
388
389 void MotorsSetup()
390 {
391            ctimer_config_t config;
392            uint32_t srcClock_Hz;
393            srcClock_Hz = CLOCK_GetFreq(kCLOCK_BusClk);
394
395
396
397            CTIMER_GetDefaultConfig(&config);
398
399
400            CTIMER_Init(CTIMER, &config);
401            CTIMER_Init(CTIMER1, &config);
402            CTIMER_Init(CTIMER2, &config);
403            CTIMER_Init(CTIMER3, &config);
404
405            CTIMER_SetupPwm(CTIMER,LM0,0,20000,srcClock_Hz,NULL);
```

```
407             CTIMER_SetupPwm(CTIMER,RM0,0,20000,srcClock_Hz,NULL);
408             CTIMER_SetupPwm(CTIMER1,RM1,0,20000,srcClock_Hz,NULL);
409             CTIMER_StartTimer(CTIMER);
410             CTIMER_StartTimer(CTIMER1);
411 }
412
413
414 void Move(int speed)
415 {
416         CTIMER_UpdatePwmDutycycle(CTIMER, LM0, speed);
417         CTIMER_UpdatePwmDutycycle(CTIMER, LM1, 0);
418         CTIMER_UpdatePwmDutycycle(CTIMER, RM0, speed);
419         CTIMER_UpdatePwmDutycycle(CTIMER1, RM1, 0);
420
421
422 }
423
424
425 void Turn_SlowLeft()
426 {
427         CTIMER_UpdatePwmDutycycle(CTIMER, LM0, 0);
428         CTIMER_UpdatePwmDutycycle(CTIMER, LM1, 0);
429         CTIMER_UpdatePwmDutycycle(CTIMER, RM0, 70);
430         CTIMER_UpdatePwmDutycycle(CTIMER1, RM1, 0);
431
432 }
433
434
435 void Turn_SlowRight()
436 {
437         CTIMER_UpdatePwmDutycycle(CTIMER, LM0, 70);
438         CTIMER_UpdatePwmDutycycle(CTIMER, LM1, 0);
439         CTIMER_UpdatePwmDutycycle(CTIMER, RM0, 0);
440         CTIMER_UpdatePwmDutycycle(CTIMER1, RM1,0);
441 }
442
443
444 void Turn_Left()
445 {
446         CTIMER_UpdatePwmDutycycle(CTIMER, LM0, 0);
447         CTIMER_UpdatePwmDutycycle(CTIMER, LM1, 0);
448         CTIMER_UpdatePwmDutycycle(CTIMER, RM0,90);
449         CTIMER_UpdatePwmDutycycle(CTIMER1, RM1,0);
450 }
451
452
453
454 void Turn_Right()
455
456 {
457         CTIMER_UpdatePwmDutycycle(CTIMER, LM0, 90);
458         CTIMER_UpdatePwmDutycycle(CTIMER, LM1, 0);
459         CTIMER_UpdatePwmDutycycle(CTIMER, RM0, 0);
460         CTIMER_UpdatePwmDutycycle(CTIMER1, RM1,0);
461
462 }
463
464
465
466 void Stop()
```

```
{
	CTIMER_UpdatePwmDutycycle(CTIMER, LM0,0);
	CTIMER_UpdatePwmDutycycle(CTIMER, LM1,0);
	CTIMER_UpdatePwmDutycycle(CTIMER, RM0,0);
	CTIMER_UpdatePwmDutycycle(CTIMER1,RM1,0);
}



void Reverse()
{
	CTIMER_UpdatePwmDutycycle(CTIMER, LM0, 0);
	CTIMER_UpdatePwmDutycycle(CTIMER, LM1, 80);
	CTIMER_UpdatePwmDutycycle(CTIMER, RM0, 0);
	CTIMER_UpdatePwmDutycycle(CTIMER1,RM1, 80);

}



void Search()
{
    CTIMER_UpdatePwmDutycycle(CTIMER, LM0, 75);
    CTIMER UpdatePwmDutycycle(CTIMER, LM1, 0);
    CTIMER_UpdatePwmDutycycle(CTIMER, RM0, 0);
    CTIMER_UpdatePwmDutycycle(CTIMER1,RM1, 75);
    vTaskDelay(750);

}
void Circle()
{
   CTIMER_UpdatePwmDutycycle(CTIMER, LM0, 90);
   CTIMER_UpdatePwmDutycycle(CTIMER, LM1, 0);
   CTIMER_UpdatePwmDutycycle(CTIMER, RM0, 70);
   CTIMER_UpdatePwmDutycycle(CTIMER1,RM1, 0);
   vTaskDelay(1000);

}



float Front_Obstarcle()
{

	float Front time,Front distance;
	GPIO_PinWrite(BOARD_Front_trig_GPIO,BOARD_Front_trig_PORT,BOARD_Front_trig_PIN,1);
	vTaskDelay(10);
	GPIO_PinWrite(BOARD_Front_trig_GPIO,BOARD_Front_trig_PORT,BOARD_Front_trig_PIN,0);

	while(GPIO_PinRead(BOARD_Front_echo_GPIO,BOARD_Front_echo_PORT,BOARD_Front_echo_PIN)==0);

	CTIMER_StartTimer(CTIMER2);

	while(GPIO_PinRead(BOARD_Front_echo_GPIO,BOARD_Front_echo_PORT,BOARD_Front_echo_PIN)==1);
	CTIMER_StopTimer(CTIMER2);

	Front_time= CTIMER_GetTimerCountValue(CTIMER2);

```

```
527         Front distance =(0.0343*(Front time/96))/2;
528
529         CTIMER_Reset(CTIMER2);
530
531         return Front_distance;
532
533 }
534
535
536 float Rear_Obstarcle()
537 {       float Rear_time,Rear_distance;
538
539         GPIO_PinWrite(BOARD_Rear_trig_GPIO,BOARD_Rear_trig_PORT,BOARD_Rear_trig_PIN,1);
540         vTaskDelay(10);
541         GPIO_PinWrite(BOARD_Rear_trig_GPIO,BOARD_Rear_trig_PORT,BOARD_Rear_trig_PIN,0);
542
543         while(GPIO_PinRead(BOARD_Rear_echo_GPIO,BOARD_Rear_echo_PORT,BOARD_Rear_echo_PIN)==0);
544
545         CTIMER_StartTimer(CTIMER3);
546
547         while(GPIO_PinRead(BOARD_Rear_echo_GPIO,BOARD_Rear_echo_PORT,BOARD_Rear_echo_PIN)==1);
548         CTIMER StopTimer(CTIMER3);
549
550         Rear time= CTIMER GetTimerCountValue(CTIMER3);
551         Rear_time=Rear_time/96;
552         Rear_distance =(0.0343*Rear_time)/2;
553
554         CTIMER_Reset(CTIMER3);
555
556
557         return Rear_distance;
558 }
559
```

Pocket Beagle Code:

```
#include <iostream>
#include"Pixy2BBB.h"
#include"uart.h"
#include"TPixy2.h"
using namespace std;

Uart lpc_link;
Uart Bluetooth;
Pixy2BBB pixy;


int x;
int y;
int sig;
int x_min=70;
int x_max=200;
unsigned int maxArea=8000;
unsigned int minArea=1000;
unsigned int width;
unsigned int height;
unsigned int area;
unsigned int newarea;
int i=0;
//uint16 t blocks;
int mySig, Sig_Status=0, Mode_Status=0;

/* Characters used for data storage */

unsigned char bt data,Mode='n',Color='n',Manual inst,j,Track Mode=1,Find=1;



int Track(char tsig)
{

       pixy.ccc.getBlocks();  //receive data from pixy

           if(pixy.ccc.numBlocks)
          {
               usleep(10000);
               if(Find==1)
               {
                   lpc_link.send("S");
                   Find=0;
               }
              sig = pixy.ccc.blocks[i].m_signature;    //get object's signature
              x = pixy.ccc.blocks[i].m_x;                    //get x position
              y = pixy.ccc.blocks[i].m_y;                    //get y position
              width = pixy.ccc.blocks[i].m width;            //get width
              height = pixy.ccc.blocks[i].m_height;          //get height
            //  printf("sig = %d    x= %d y= %d  width = %d height= %d \n
  area=%d",sig,x,y,width,height,width*height);



              if(sig==tsig)
              {
                    newarea= width * height;

                    printf("newarea %d\n",newarea);

                    if(x<x_min )
                    {
                        lpc_link.send("L");                        //Send command to lpc to turn left
```

```
                        //printf("Left\n");
                        usleep(100000);

                    }
                    if(x>x_max )
                    {
                        lpc_link.send("R");                         //Send command to lpc to turn right
                        //printf("Right\n");
                        usleep(100000);
                    }
                    if(newarea<7000)
                    {
                        lpc_link.send("M");                         //Send command to lpc to Move
                        //printf("move\n");
                        usleep(100000);
                    }
                    else if (newarea>maxArea)
                    {
                        lpc_link.send("B");                         //Send command to lpc to Move in reverse
  direction
                        //printf("back\n");
                        usleep(100000);
                    }
                 else
                {
                    lpc_link.send("S");                             //Send command to lpc to Stop
                    printf("stop\n");
                    usleep(300000);
                }
            }}
        else
        {
            usleep(300000);
            lpc_link.send("F");                                     //Send command to lpc to Find object
            usleep(300000);
            Find=1;
        }

    return 0;
}


int main()
{

    pixy.init();                                                    //Initializing Pixy2


    lpc_link.Init(UART04,115200);                                   //Initializing Uart
    Bluetooth.Init(UART01,9600);                                    //Initializing Uart


begining:
    pixy.setLamp(1,0);
    Mode_Status=0;
    Sig_Status=0;
    lpc_link.send("S");
    Bluetooth.send("Welcome\n");
    usleep(900000);
    Bluetooth.send("Please Select the mode \n");
    usleep(900000);
    Bluetooth.send("A for automatic and M for manual \n");
    usleep(100000);
    pixy.setLamp(0,0);

    while(Mode_Status==0)
    {
        while(Bluetooth.recieve(&Mode)<2);
```

```
144         if(Mode=='A')
145         {
146             Bluetooth.send("Please Select the color to track \n");
147             usleep(900000);
148             Bluetooth.send("Options- P for Purple Ball \n G for Green Ball \n");
149             usleep(900000);
150 
151             while(Sig_Status==0)
152             {
153 
154                 usleep(1000);
155                 while(Bluetooth.recieve(&Color)<2);
156                 printf("%d\n",p);
157                 pixy.setLamp(0,0);
158                 if(Color=='P')
159                 {
160                     mySig=1;
161                     Sig_Status=1;
162                     Mode_Status=1;
163                 }
164 
165                 else if(Color=='Q')
166                 {
167                     goto begining;
168                 }
169                 else
170                 {
171                     Bluetooth.send("Enter a valid Option\n");
172                     //Sig_Status=0;
173                 }
174             }
175 
176         }
177 
178         if(Mode=='M')
179         {
180             Bluetooth.send("Manual Mode\n");
181             while(1)
182             {
183                 Bluetooth.recieve(&Manual_inst);
184                 if(Manual_inst=='M')
185                 {
186                     lpc_link.send("M");
187                     usleep(400000);
188                     //printf("ffff");
189                 }
190                 else if(Manual_inst=='B')
191                     {
192                     lpc_link.send("B");
193                     usleep(400000);
194                             }
195                 else if(Manual_inst=='L')
196                 {
197                     lpc_link.send("L");
198                     usleep(400000);
199                 }
200                 else if(Manual_inst=='R')
201                 {
202                     lpc_link.send("R");
203                     usleep(400000);
204 
205                 }
206                 else if(Manual_inst=='S')
207                 {
208                     lpc_link.send("S");
209                     usleep(400000);
210                 }
211                 else if(Manual_inst=='F')
212                 {
213                     lpc_link.send("F");
214                     usleep(400000);
```

```
                                }
                else if(Manual_inst=='Q')
                {
                    goto begining;
                }
                usleep(200);
            }
        }
        else if(Mode=='Q')
            {
                goto begining;
            }

        else
            {
                Bluetooth.send("Please Choose an valid option \n");
            }


    }


    Bluetooth.send("Tracking \n");
    //pixy.setLamp(1,0);
    while(1)
    {

        Bluetooth.recieve(&j);

        Track(mySig);
        if(j=='Q')
        {
            lpc_link.send("S");
            usleep(30000);
            goto begining;
        }

    }


return 0;
}

```

Working: This code receives the data from the pixy and it is also responsible for communication with UART and Bluetooth module it receives the input from user then run the pixy to track the object and communicate with another half brain that is LPC54114. It drives the robot. It can be considered as master brain code. It is a master code without this code the robot will not move.

Demo Day Experience

Demo day was held on August 22, 2019. It was very exciting for us after so much hard work; we finally completed our robot, which was fully functional with different behavior. Hardware and software were working perfectly without any issues. Our robot was tracking object with great response time without any delay. Everybody had their own robot in class. It was like a group event of different robots with different behavior. We named our Robot PIXYBOT; everybody had to demonstrate their project and its functioning. We went first for the demonstration our instructor asked us few questions about the hardware designing, about the problem faced during the project and our code. He gave his expert advice by studying our code which helped us to improve the code and overall performance of our Robot. For example, we got to know that it is always better to avoid go to statement. We reduced the clock speed of LPC54114 so that we could save power and the runtime of our robot increased. We really enjoyed the day and getting suggestions from respected instructor and friends to improve our PIXYBOT. Last but not least our professor gave us advice on what things to focus on in future so that we could have good opportunities in jobs.

Future Plan

Building this project was a wonderful experience. We got to learn a lot about Embedded c, FreeRTOS, Linux, C++, C. The process was quite clear we started from planning the components, designing Hardware, implementing hardware in best way possible, then working on our code. It is really interesting when you know your code is going to affect the hardware, so we had to be very careful writing it. We got to learn about two new MCU, s LPC54114 and Pocket Beagle. We believe that our exposure to freeRTOS, embedded c, and C++ will help us to get good opportunities in the market. This robot is a great platform it has its own vision system, two brains, movement everything. We believe that building this robot demonstrates our embedded c, C++, hardware designing, planning, implementation, and many other skills. It will definitely help us in future for research purpose as well as getting good opportunities so that we can grow and learn more and build something to make this world better.

APPENDIX

Website links:

Ankit jhall: ankitjhall.github.io

Jaspreet Singh: jasskhaira.github.io

GitHub links:

https://github.com/ankitjhall

https://github.com/jasskhaira

YouTube links:

https://youtu.be/D_ZXO4Isk0I

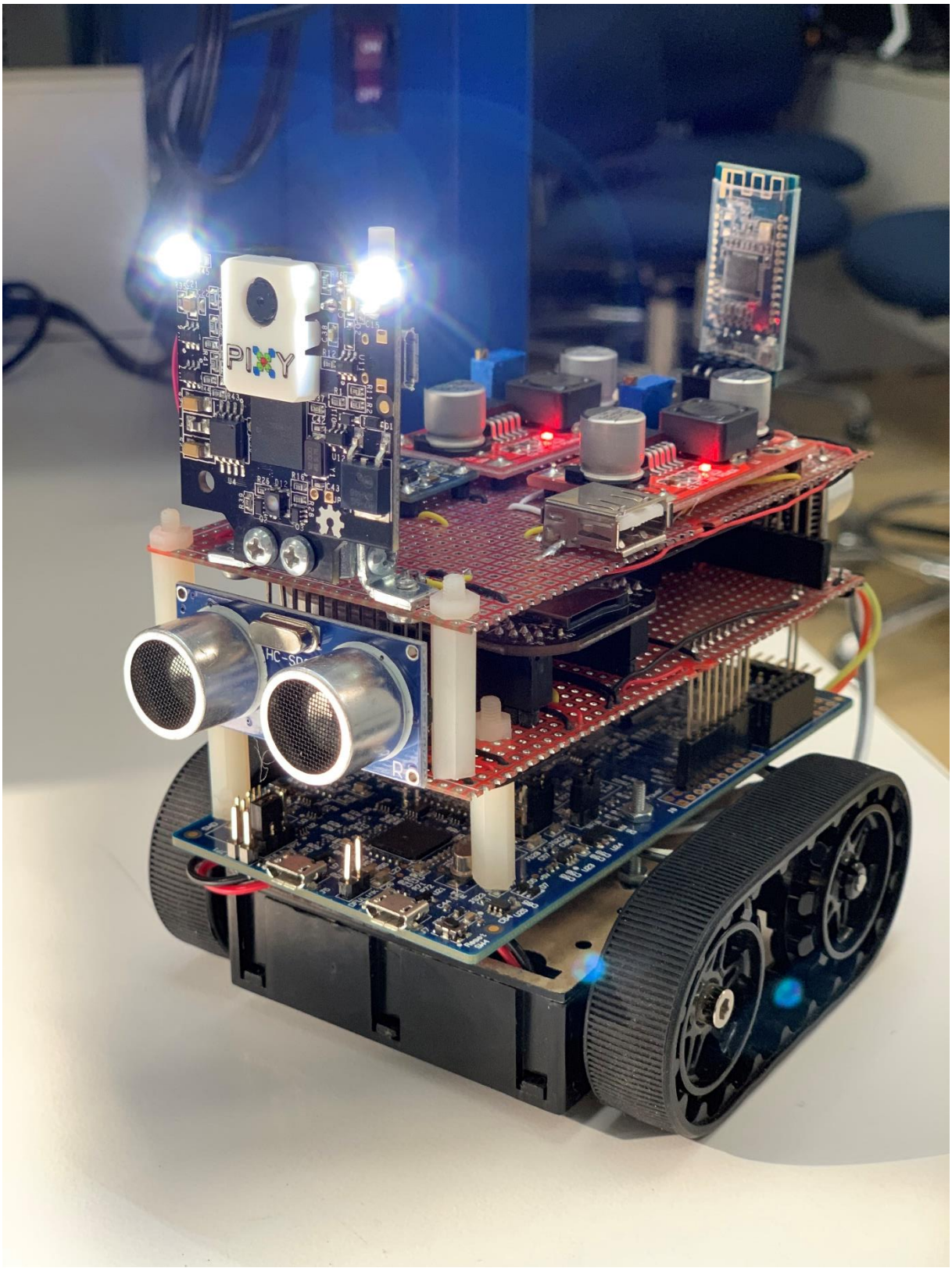
https://youtu.be/fWj-_V181j8

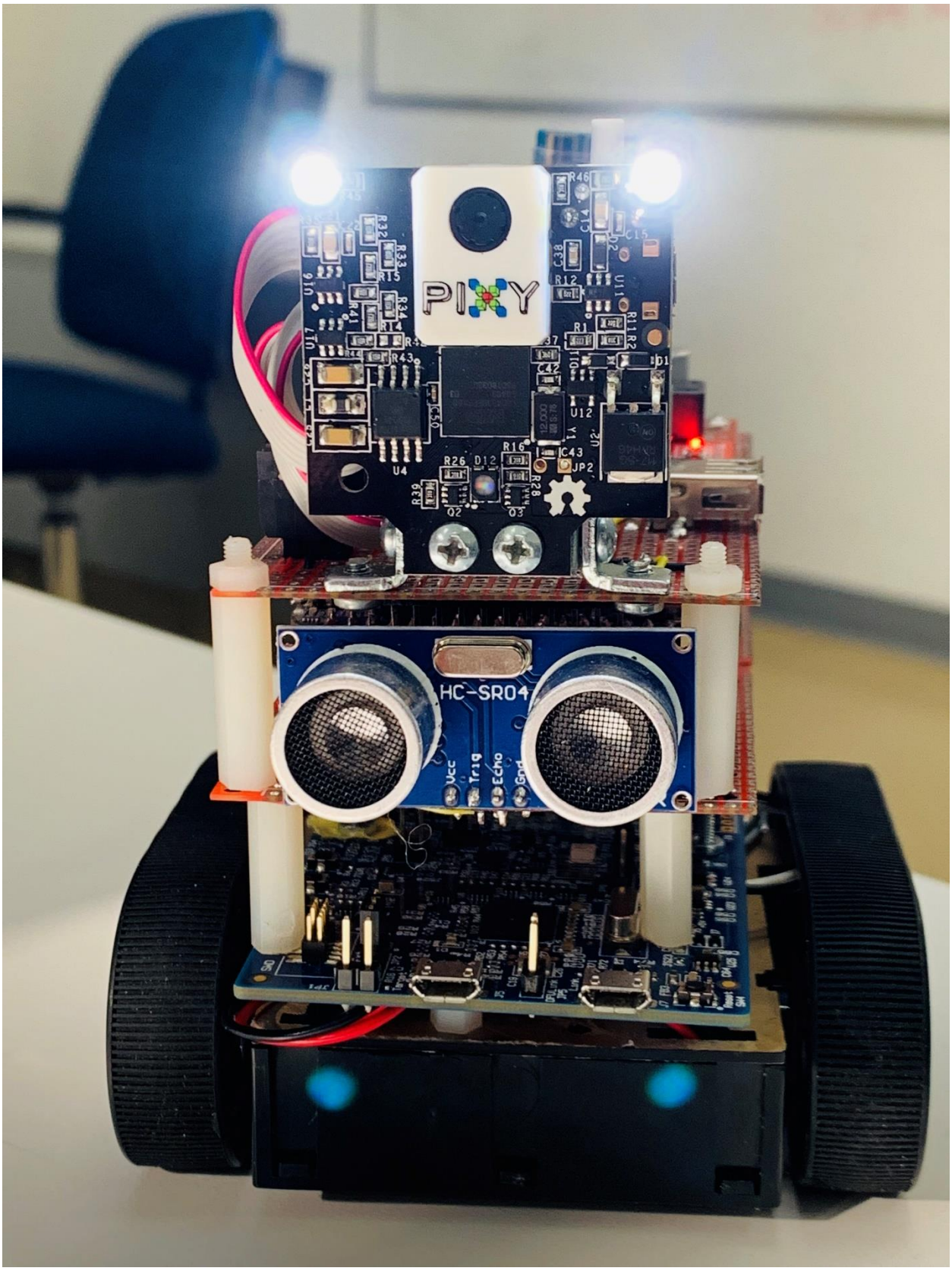
https://youtu.be/T_iYn4tgbec

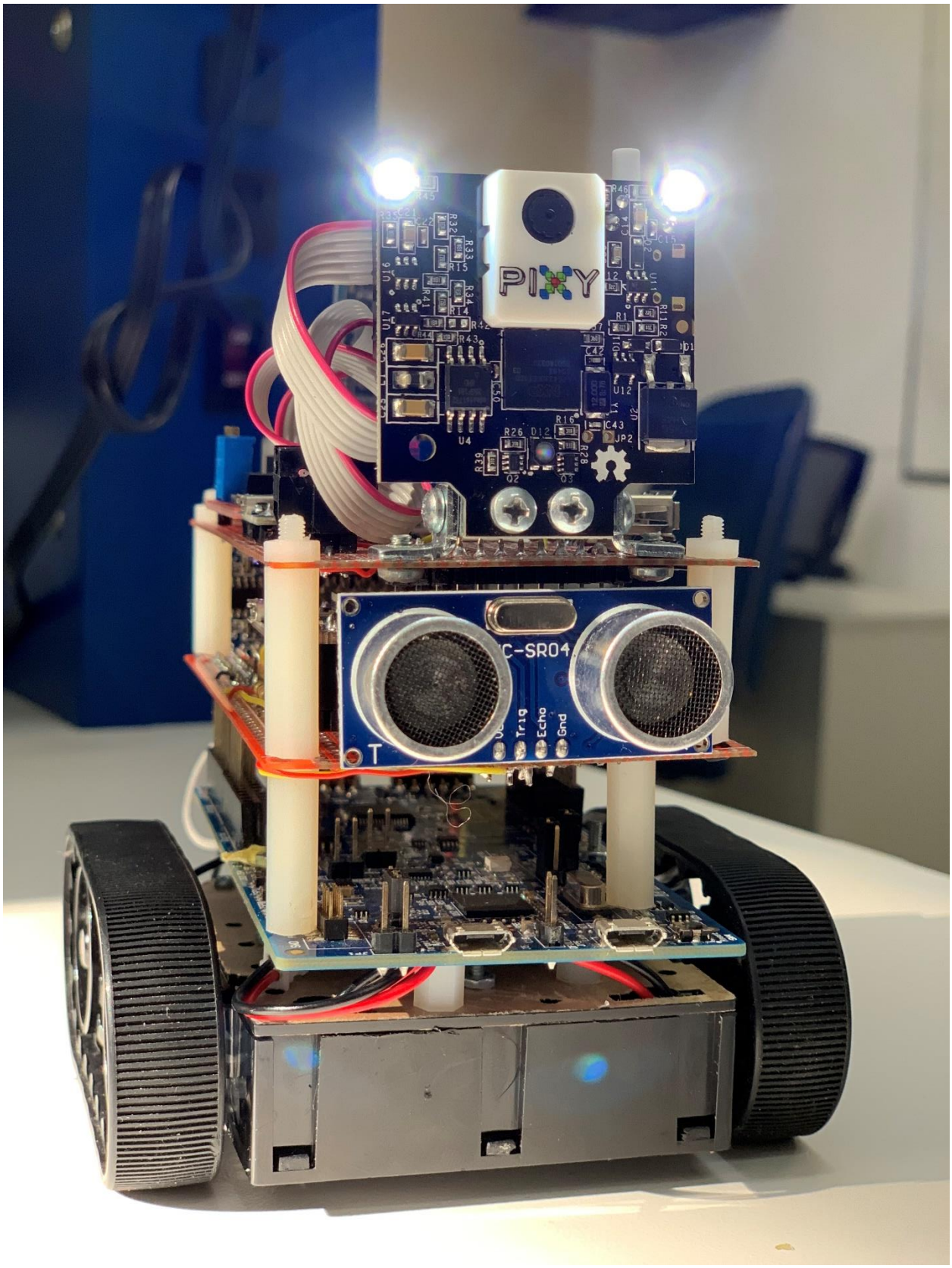
Visit GitHub links for the CPP file of code.

PDF files are also available on GitHub.

Images of PIXYbot are shown below


PIXY


PIXY
HC-SR04
Vcc
Trig
Echo
Gnd


PIXY
Trig
Echo
Gnd

References

Beaglebone. (n.d.). *beagleboard*. Retrieved from beagleboard: https://beagleboard.org/pocket

NXP LPC54114. (n.d.). Retrieved from os.mbed: https://os.mbed.com/platforms/LPCXpresso54114/

NXP. (n.d.). *Nxp Products*. Retrieved from NXP: https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/lpc54000-cortex-m4-/low-power-microcontrollers-mcus-based-on-arm-cortex-m4-cores-with-optional-cortex-m0-plus-co-processor:LPC541XX

PIXY. (n.d.). Retrieved from pixycam: https://pixycam.com/pixy2/