Making contact sensors for a robotic model of a cricket

Ehsan Honary

MSc in Artificial Intelligence
Division of Informatics
University of Edinburgh
2000

Abstract

This project is part of the ongoing research for making a robotic model of a cricket. The Koala robot from K-team is used as a base for the sensors. There are a few sensors that are currently developed. Sound seeking [1] and optomotor sensors [2] are already made. The idea is to use the Koala robot outdoors as a model of a cricket. This project is about making artificial contact sensors. These would collect information about the amount of pressure applied as well as the direction of impact. In this project possible designs for these sensors are explored, the sensors are made, and then different algorithms are implemented to use them.

Acknowledgements

First of all I would like to thank my supervisor, Dr John Hallam for being a great guide in this project helping me to recover from the difficult parts with confidence. Also thanks for the proofreading of the dissertation and providing helpful suggestions, to make this become a better story for the reader.

I would also like to thank Rob Macgregor for fabricating PCBs and various other mechanical and electronic requirements for this project. Also thanks to Sandy and other members of the workshop for providing help when I needed it.

Special thanks to Richard Reeve, my second supervisor, for setting up the goals and also helping me in various times to go towards the right directions.

Thanks to Jose Lopez and other classmates for making the MSc course a wonderful year.

And I would like to thank my parents for their great support during this year and their great advice and recommendations in those moments when I felt I need it most.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This project is about making sensors. The robot used for this project is the Koala robot from K-team which is largely backward compatible with their successful Khepera robots. The most important feature of Koala is that it can be used outdoors. This is an important feature since this project is part of a bigger goal which is to make a robotic model of a cricket and be able to use it outdoors for future research.

1.1 What is Koala?

The Koala robot is a relatively big robot, designed to be of practical use compared to Khepera, but still with some limitations for being used outdoors. For example, the robot is not waterproof! But on the other hand it has six wheels and has a sporty build and look. Fig 1.1 is a picture of the robot when it is bought off the shelf.

Some of the important specifications of the robot are that it uses a Motorola 68331 processor at 22MHz, has 1MB RAM and 1MB FLASH, 2 DC motors and 16 infra red sensors. It uses NiCd or NiMH batteries which last around 2 to 6 hours depending on configuration. It has 12 digital inputs, 6 analog inputs, and some other special purpose outputs for connection to the

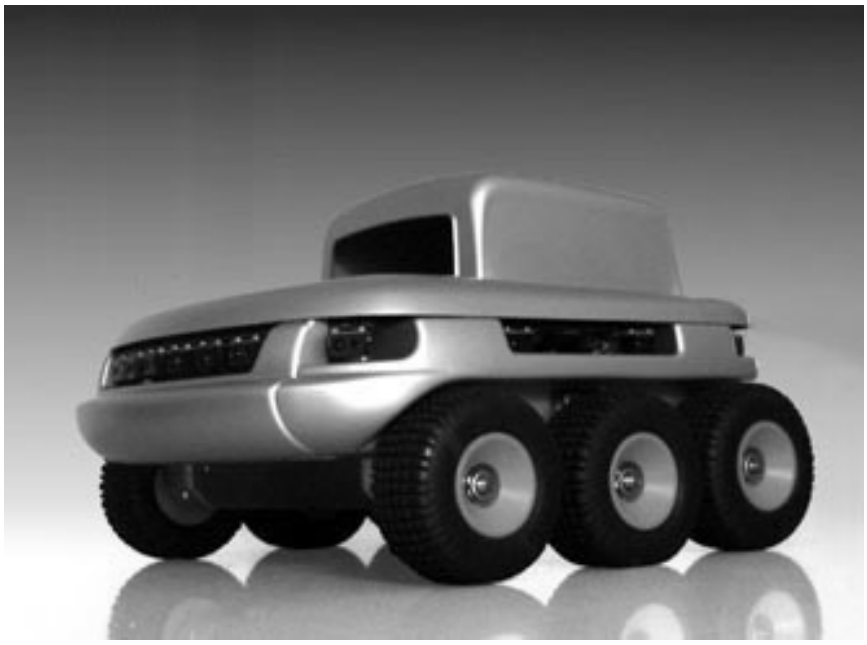

Figure 1.1: The Koala robot. Length: 32 cm. Width: 32cm. Height: 20cm. Weight ≈ 3Kg.

outside world. See Fig A.2 or [9] for more information.

The robot can be used in various ways using different modes. It can be attached to external processing, like a computer, through serial communications. It can work stand-alone, by down-loading the software into the FLASH memory on board. It is also possible to attach a mini-PC card such as PC104 through another serial communication port.

There are two main motors available, to drive three wheels on each side. The Koala main processor has the direct control over the motors and this is controllable through BIOS[1] commands supplied by manufacturer. These are routines made by the manufacturers so the users can interact with the system resources with no need to know the underlying design. It is possible to use assembly or C language if software is down-loaded into the Koala. Of course, if an external processor is used, the choice is totally open, since all

[1]Basic Input Output System

it needs to do is to communicate through RS-232 serial communication with the Koala.

1.2 What sensor to make?

First consideration was to choose a particular type of sensor to make. A list of possible ideas are:

- Hair sensors that detect the direction of the wind. They could have strands of wire that would connect to directional plates on the base and work like a switch to indicate from which direction the wind is coming. Using this, it is then possible to create an algorithm to make the robot move towards the direction of the wind or to run away from predators, which is what cockroaches use wind sensors to trigger. Some work is already done on this by Tim Chapman [5].

- Antennae in front of the robot to detect obstacles. This is currently under development by Graeme Watson [3].

- A tilt sensor to detect the tilt of the robot in different situations. Since the robot is going to be used outdoors, it is always useful information to know what angle the robot was relative to the ground. This could then be used to avoid going to a dangerous edge where the robot can get stuck because the wheels are no longer touching the ground.

- Accelerometers could be used to tell the robot about the acceleration or deceleration of robot in certain situations. If it commands the motors to go forward and the accelerometer does not read anything, then the robot must be stuck. It is possible to design a specific sensor for this, but the speedometer of the robot could also be used for the same purpose.

- Pressure sensitive sensors that could be used all around the robot to detect bumping into something when the robot fails to detect the obstacle by using some other sensors. These could also be used to detect if something runs into the robot instead of the robot running into something.

Considering all these options, the last one, to make pressure sensitive sensors was chosen for this project. After all, the crickets have an exoskeleton and can detect pressure and it was felt it would be sensible to include this in the overall project as well as being able to make the robot more practical when it is used outdoors.

1.3 Contact in biological animals

Before discussing how might a pressure sensitive sensor be made, it is a good idea to know how this is done in nature. Human skin is well studied and would serve a good start to see how touch sensitive sensors might be made. Human skin, as illustrated in Fig 1.2, serves many diverse functions. The skin consists of many layers which are regenerated all the time and the outer ones are basically dead cells. There are various sensitive groups of cells underneath the skin such as pain sensors, hot and cold sensors and touch sensors.

There are several kinds of touch organs. One kind of touch organ is found near hairs, another kind in hairless areas, and still another kind in deeper tissues. The sensation occurs when an object comes in contact with the sense organs and presses them out of shape, or touches a nearby hair. Nerves from the organs then carry nerve impulses to the brain [12].

Touch is more sensitive in some parts than in others. This difference is due to the fact that the end organs for touch are not scattered evenly over the body, but are arranged in clusters. The feeling of pressure is most sensitive

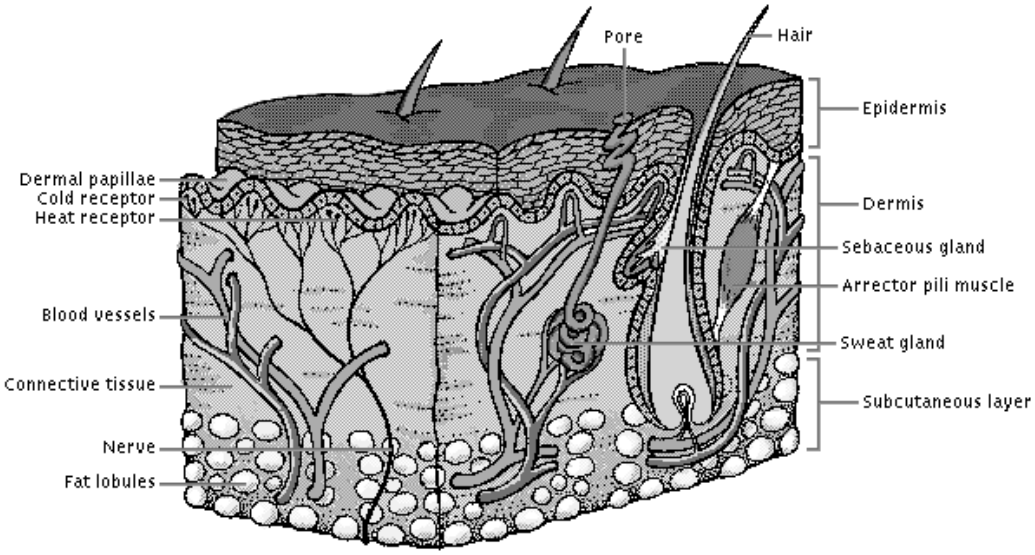

Figure 1.2: Human skin. Picture taken from encyclopedia Encarta.

where there are the greatest number of end nerves. The skin material has suitable friction characteristics which make it stick more easily to materials and so it deforms easier [12].

In comparison, insects are more sensitive to touch than people are. The touch organs consist of hairs and spines that cover all parts of an insect's body, even its eyes. The hairs are especially plentiful on the antennae. Any kind of pressure moves the hairs, setting up a nerve reaction that goes to the brain. Some of the hairs are so sensitive that the gentlest air current can bend them [12].

1.4 First thoughts

Using analogy between biological skin and the mechanical world, it is possible to come up with an initial design. The first consideration for the development of the sensors is the kind of material to use to detect the pressure on impact. Mechanical methods such as use of moving bars and switches have already

been tried and they have certain limitations. For example they usually require clever mechanical designs and also are still subject to blind spots, a characteristic to be avoided.

So something that works similar to skin would be the ideal. It should be flexible so it could be attached around the robot, without much difficulty, and should also be sensitive enough to detect various kinds of bumps and also strong enough so it would not get damaged. It should also be easy to get position information from this sensor, so it would be possible to detect where the robot has been hit.

PVDF films were chosen since they are becoming widely available and are also used in a variety of applications. They are flexible, they could be used all around the robot easily and they come in all sorts of shapes and sizes. It is possible to use different patches of the sensor to detect the position of the impact.

1.5 What is PVDF?

An introductory explanation for PVDF is presented here, but more details on getting the "right" signals out of PVDF are presented in relevant chapters later on.

Polyvinylidene fluoride (PVDF) is a semi-crystalline, high molecular weight polymer of repeat unit $CH_2 - CF_2$ whose structure is essentially head to tail. As it would be described later, we use Kynar^{TM} piezo films, which are produced by Atochem.

PVDF is basically a piezoelectric material. This means that upon impact the film will generate a certain amount of signal, which then could be amplified to be used later on. The film is like a charge generator with a capacitor in parallel. Piezo films also act as a pyroelectric transducer which can be used to detect thermal radiation [7], but this particular aspect of them is not

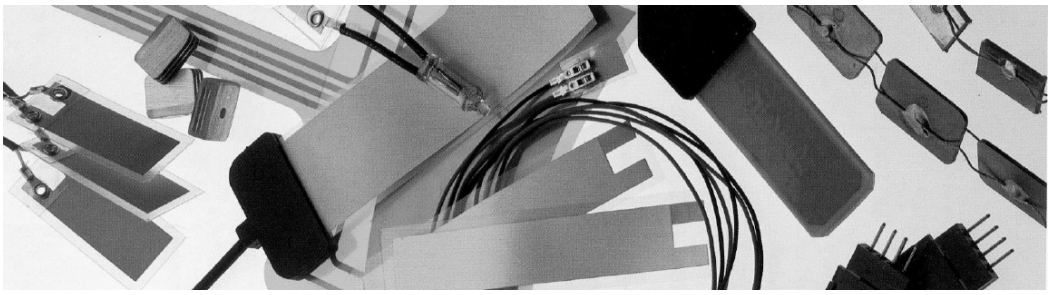

Figure 1.3: Some samples of PVDF films. Picture taken from Kynar Technical manual, Atochem Inc.

used in this project.

Piezo films generate electrical charges that are proportional to the amount of mechanical stress applied to them. PVDF film does not require external power sources and generates relatively strong signals. In the simplest mode it works as a strain gauge and the low thickness of the film can make it quite sensitive.

The way that the film works could be illustrated with an analogy. The film behaves like a sponge with water in it. When the sponge is squeezed, the water comes out of it and when the pressure is gone the water goes back into it. So the film under each impact would generate a positive and negative signal as charge is first expelled and then recaptured. The speed of the pressure applied is also important. For example if a pressure is applied slowly, the film starts bending until is maximally bent. But the signal produced is not as much as when the same displacement is made by a stronger and faster impact. Fig 1.3 shows some of the possible shapes for this film.

1.6 Summary

So, by now the components to be used in this project are introduced. Considering some of the background thinking on skin and bumper design, it is time to put these together and see what can be achieved. This is presented in the next chapter.

Chapter 2

A Draft Of The Problem

In this chapter the requirements of the project and possible problems which will be encountered are discussed. In later chapters, each issue is thoroughly examined and possible solutions are presented.

2.1 General considerations

The next step is to have a design for using PVDF as the sensors. PVDF sensors generate an electronic signal which might need to be amplified. Then the amplified signal should be converted to some other sort so it could be sent to the Koala robot. The Koala should be able to decode this message in real time and be able to use it fast enough so it can avoid bad bumps or simply be able to perform some useful behaviour. There are going to be sensors all around the robot so it should be able to detect bumps from every side without having any blind spots.

For easier development and fabrication, the sensors could be made as separate modules. Each can have separate electronics with different kinds and sizes of PVDF and other necessary materials attached.

2.2 Mechanical requirements

For a start, the sensors should withstand impact. They have to be strong enough so bump after bump they would generate a proper signal within the specification which should not change from impact to impact. They also, as any engineering application, have to be cost effective. As described before the Koala is relatively big and, since it is going to be used outdoors, possibly gardens, should be able to withstand bumping into irregular objects such as branches and pavements.

PVDF needs to absorb the impact so it can generate a better signal. So another requirement is to mount the sensors in such a way that they can get the maximal impact without getting damaged. An idea is to put them vertically around the robot with some protective sponge.

The robot should also be waterproof. However the original designers of the Koala robot have not considered this, and some modifications should be done to at least make it splash-proof.

The wiring around the robot should also be minimised, since in the future other sensors are going to be added and it is not desired to have a mess of wires around and is also better to make the robot lighter for a longer battery life.

2.3 Electrical considerations

There are various requirements on electronic side. First it should be possible to get a reasonably "powerful" signal out of PVDF. This could be achieved by some sort of amplification. In order to get the best results, the electronics circuit has to be able to distinguish between the noise and the signal. The noise is generated by sources such as vibration caused by the robot, noise picked up by the wires, EMI or electro-magnetic interference caused by other electronic equipment, different characteristics of electronic components, loud

sound and many others.

Then this signal can be converted to digital for sending to the Koala. The communication has to be fast enough so the Koala can respond within reasonable time to impacts. On the other hand there is always a limit to the speed of communication. So the best design is the one that has the optimum speed of communication and is also cheap.

If there is going to be a signal for each module or sensor, then the number of these are also limited. As described before there are only 12 digital inputs available on the Koala robot and since other sensors are going to be added, saving inputs is always a good idea.

The robot works with batteries, so the power consumption by the electronics of the sensors has to be minimised as much as possible. It is also better to use one single PCB[1] design for each module around the robot, to cut the costs and development time. Since the circuit for modules will be the same, even slight modification would save some valuable current.

And also as usual, the fewer components the better the design. This will usually cut the cost in fabrication of the PCBs and also would make them smaller, lighter and more elegant.

2.4 Can the Koala handle it?

The speed of the processor on the Koala is limited (only 22MHz). The program used for it is going to be C language using the BIOS functions provided from the manufacturer. This will impose a certain amount of inflexibility in the design of software.

It is unfortunately not possible to use full features of interrupts available from the processor for timing signals. Some time sharing routines are provided, which could be useful. However reading the timing of signals in a

[1]Printed Circuit Board

timesharing environment without using interrupts is a difficult task.

2.5 Summary

In this chapter the problem and the requirements were described and it is now time to present the possible solutions to each category and then select the best. In designing the sensors at any stage, most of the other requirements should be present in the back of the mind of the designer, as they might influence the decisions to make in particular areas. However, as in any engineering project there are always trade-offs between certain requirements. Sometimes it is possible to come with a great idea that satisfies some requirements but would create problems for others. In general, the best design can be achieved by finding a "right" balance between these requirements. The hunt for this balance is the goal of this project.

Chapter 3

Mechanical Design

In this chapter, possible designs for mounting the sensors and mechanical issues are discussed.

3.1 Mechanical specification of the sensor

The PVDF comes in various shapes and sizes. The particular shape that is sensible to use, and also readily available and cheap, is rectangular. After going through some preliminary experiments with the film, some ideas emerged. The films can not be too thin. If so, they would be too insensitive and would require a lot of amplification to get a desired signal. On the other hand they should not be too thick, since thicker films produce higher outputs and if a heavy impact is applied, the result can be out of the range of amplification and hence not really suitable. For the purpose of this project 28μm thickness was used.

The films can also come in laminated form. This is basically a thicker plastic film attached to piezo film in manufacturing process. This will spread the impact on all the area of piezo film which makes it generate a stronger signal for a certain impact. In practice laminated films work better, so they were used eventually. See Fig 3.1 for samples of some PVDF films.

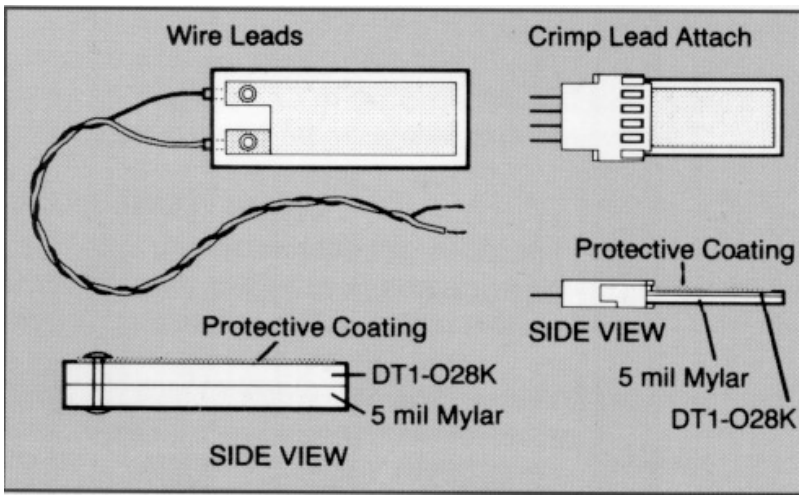

Figure 3.1: Various PVDF films. Picture taken from Kynar Technical manual, Atochem Inc.

3.2 Mounting the sensors

The PVDF films are to be mounted on the Koala, in such a way that they would have optimum position on impact to generate the "right" signal, which would represent a unique impact uniquely. The Koala robot is, unfortunately, difficult to work with mechanically. Also as discussed before, waterproofing was to be considered too. So first, a new plate was mounted under the base of the robot[1]. This now gave a platform to mount the sensors. However, this blocks the Koala's infra red sensors (IRs). So the IRs had to be moved out of their original place and be mounted on top of the plates, so they could be used later on by future researchers.

So, how many sensors are needed? A practical number is five. It is then possible to have two in front and one on each side and one at the back. It is usually better to get more information, but sometimes a lot more information might not bring a lot more efficiency. Having more modules on the sides or

[1]Mechanical design of the plates and various other things were done by Robert Macgregor and the workshop.

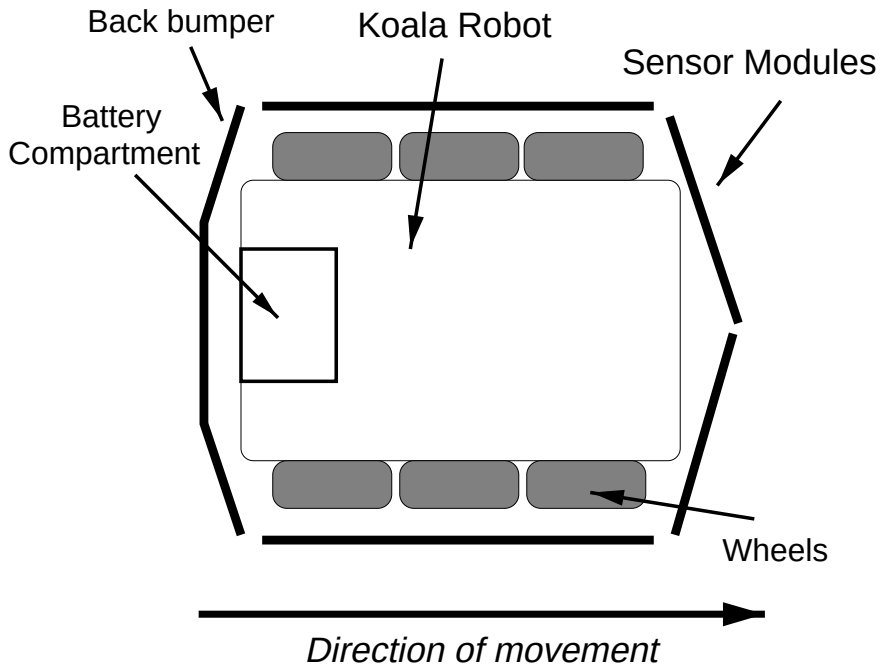

Figure 3.2: The five modules around the Koala robot.

at the back is really not necessary, since directional information might not be used as much as it is used by the front. Besides, as we shall see, it is possible to get more directional information from even a single module. So two modules in front was also considered practical. The arrangement of modules is illustrated in Fig 3.2.

In order to mount the modules, there was a need for some plates that could be mounted vertically on the edges of the main plate mounted underneath the robot. The height of these was experimented with and a practical figure was about 6cm. The modules should be able to detect most of the bumps within the same height as the robot. However if "tall" sensors (sensors that are positioned higher than the contact sensors) are added later on, some other way has to be devised such that the robot would not run into something without detecting it.

The back side of the robot was divided into three sections so it would

be possible to remove one section for changing the battery. Also the angles of the plates around the robot were adjusted in a way to give maximum chance to the robot in getting out of bad situations as fast as possible. It is always better to take advantage of the physical characteristics rather than programming for every possible situation. If the front bumpers are made with an angle, then the robot can turn much more smoothly without continuously giving signals for the impact of front bumpers.

3.3 Modules

The modules should contain certain flexible materials, such that the PVDF would be able to bend, so it can generate a signal. For this purpose using sponge was considered. By attaching PVDF on top of a soft sponge it is possible to transfer some displacement to the film upon impact. However the connections to wires on the film are fragile and if the film itself is bumped into objects all the time, it would reduce the life time of the film. For this reason a stronger sponge is used on top of the film for the outer layer. This would also spread the energy of the impact to more area of the film which can then generate stronger signal.

To choose between sponge types, there was a need to get a signal out of the PVDF film and to be able to compare it to the signals taken in different conditions. For this purpose, a standard impact was implemented. This was basically, dropping a sphere from a certain height (which was set to be 10 cm) on top of the pad or module. The PVDF film would then bend because of the impact and the voltage could be read accordingly. The sphere had a diameter of 2.5 cm and it had a weight of 55 grams.

This test was done for different film thicknesses and different PVDF film sizes as well. Different sponge material was used and the standard impact was applied on them and the signals received from PVDF films was compared.

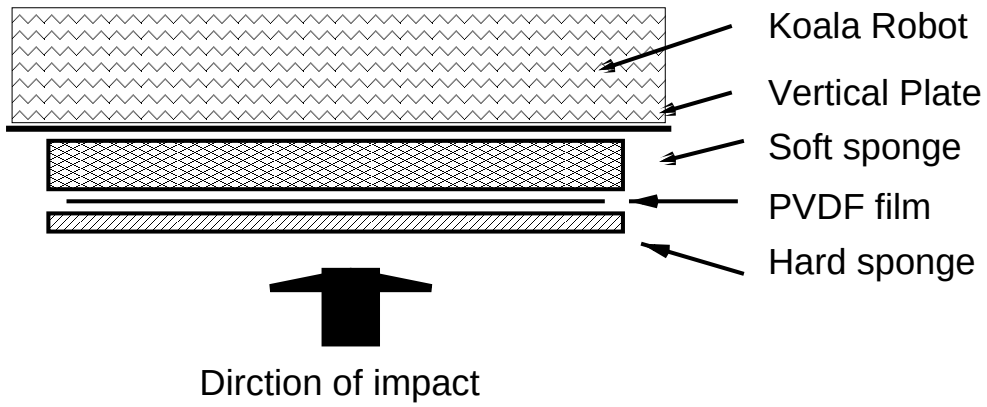

Figure 3.3: The PVDF film is sandwiched between two kinds of sponge for each module. Top view.

The idea was to choose a combination of PVDF film and sponge were the sensitivity could be high and easily adjustable. If the combination produced more voltage, it was generally a better option because the signal had a more range. This means that smaller impacts could be distinguished from each other much more easily than when the voltage was not as high.

Eventually, for the flexible type, a 1cm thick soft sponge was found to be useful and a kind of mouse pad was used for the outer layer. Putting films and sponge together is illustrated in Fig 3.3. The picture of a front module is shown in Fig 3.4.

So now, each module contains a few films, some sponge and a PCB for the electronic circuit. Each of the modules is then glued to a plate which is mounted vertically on a side of the robot.

3.4 Layouts

Each module can have different configurations for placement of the films, number of films, or different sizes of the films used. This was extensively investigated and it was considered in the design of the electronics and the

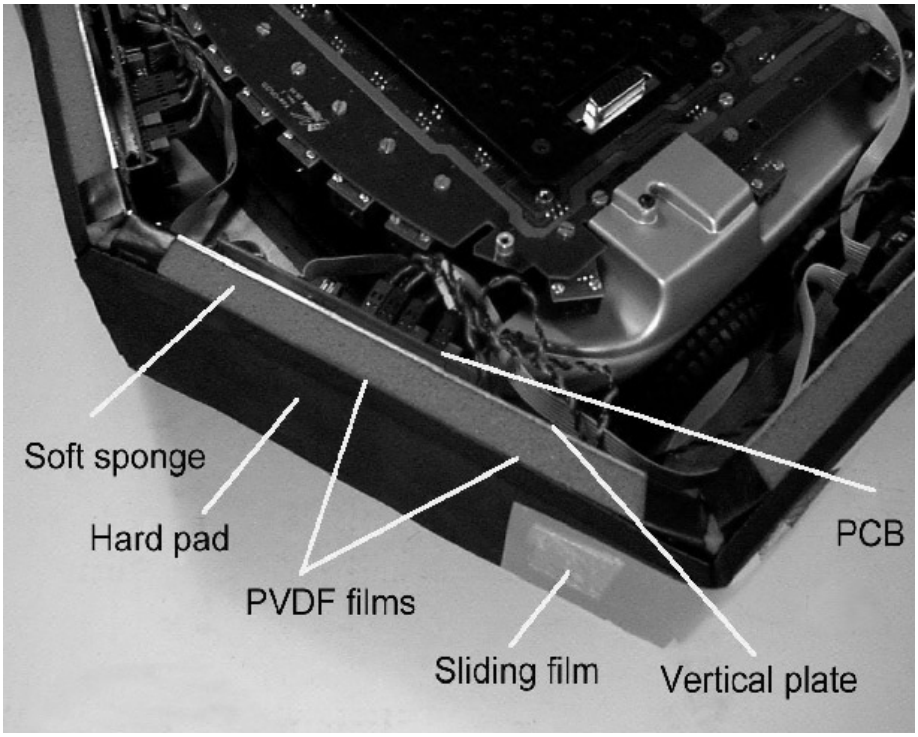

Figure 3.4: The picture of front module attached to the vertical plate. Sliding film would be explained later in chapter 8.

microcontroller software, such that it would be possible to see if the behaviour of the robot could be affected by the choice of the layouts.

For example, the front module is 6 cm by 20 cm. As described in later chapters, there could be up to three signals from each module. This is imposed by the limited number of inputs available on the microcontroller used in this project. However, it is possible to use more than three films. For example, to use four films, two of them could be connected in series to one channel, and be used as a single input. This way it is possible to cover more area of the pads, and prevent having blind spots. The layouts used in the configurations of Table 5.1 are illustrated in Fig 3.5. The modes are extensively described later on in Chapter 5.

In designing the layouts, covering more area of the pad without using

many films was one important consideration. However, sometimes more films are considered to observe if they would make any difference in behaviour. There are many kinds of films available, but only two were used for this project considering the experiments and requirements described earlier. One is LDT2-028K/L which is 16.26×72.64 mm and the other is LDT4-028K/L which is 21.84×170.69 mm. Both films come in laminated condition to give stronger signals on impacts[2]. The results of tests are described in later chapters.

3.5 Summary

In this chapter, possible solutions to mechanical problems were discussed. It is always possible to use alternative mechanical designs, and in general only the designs which take advantage of the physical world are successful. Otherwise satisfying other requirements becomes too difficult. Whether the right design has been chosen is to be seen if the remaining requirements are satisfied easily.

[2]The films are bought from Measurement Specialties from USA. Their web page is www.msiusa.com.

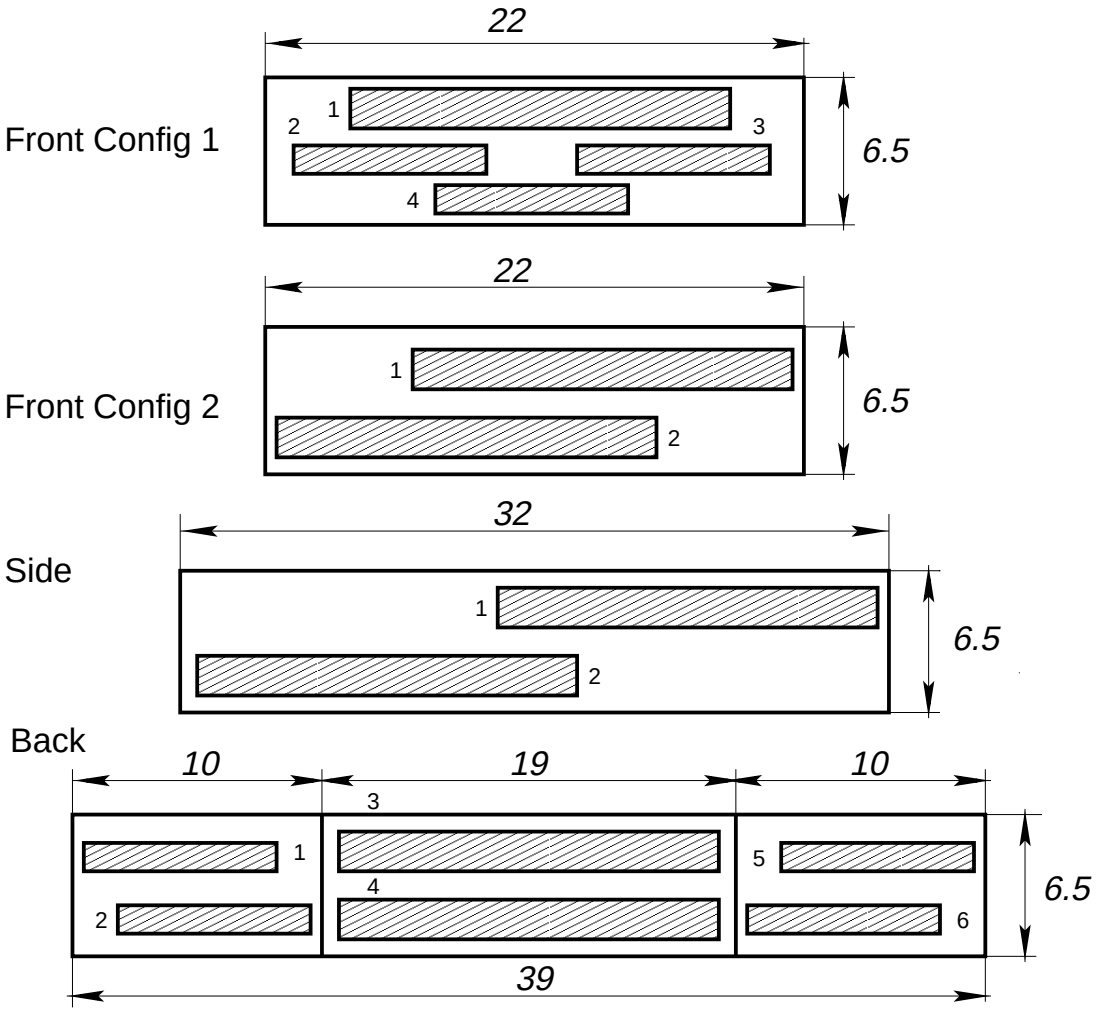

Figure 3.5: Layouts used for different configurations in multi-configuration mode. All dimensions are in cm. The front and side layouts are viewed from the back of the module. The back module layout is viewed from the front of the module. Convenience of understanding directions is the reason behind this. The numbers next to layouts are used for connecting the wires to PCBs conveniently.

Chapter 4

Electronics Design

In this chapter issues on getting a good signal from PVDF are discussed, and various points about designing the circuit digram for modules are explained.

4.1 Some ideas for electronic design

How is it possible to transfer the information to the Koala, once the analog values of PVDF films are read? What follows are a few ideas that were considered as part of the search for the final design.

One way for extracting information is to read the sensor's analog values after amplification and threshold them to have only one bit of information. Then all these lines from all the sensors, for example five, would go to a parallel to serial converter and this serial line would go to the Koala robot to be decoded and used. Notice that this way the information about the amount of pressure applied is lost. The benefit of this design is that there is only one simple electronic board. However the bad points are the information loss and possibly existence of lots of wires all over the place.

A better design based on this, is to have serial communication between all the sensors. Each sensor would be a module that would have a microcontroller which would be connected to a common serial line. This way, there

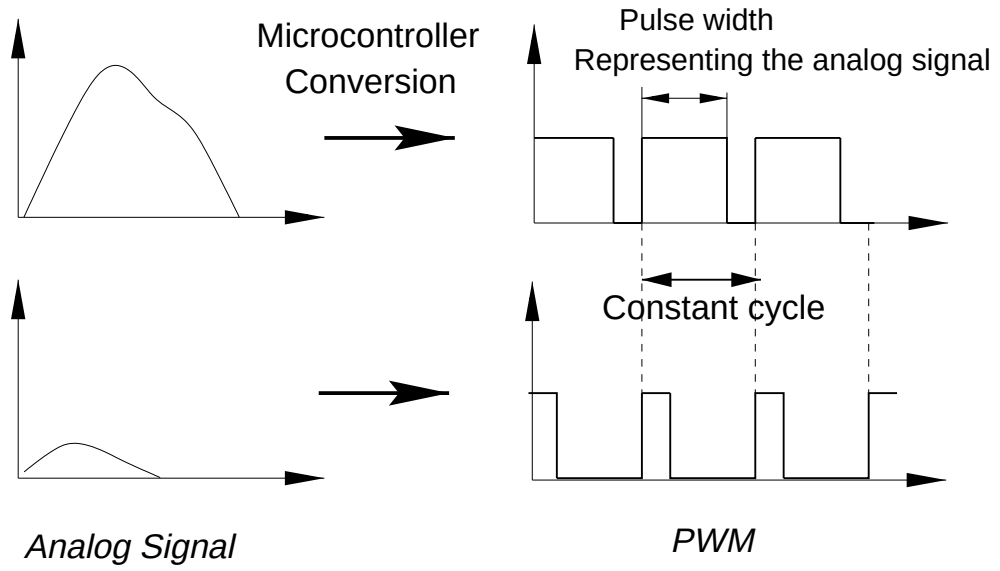

Figure 4.1: Two different levels of analog signals on the right are converted to PWM pulses shown on the left.

are only 3 wires going from each module to the next. The information is binary and there is still a loss of analog information.

There is already a neural network software developed for the Koala robot [6]. To use this to advantage, one way to pass the information is to use PWM pulses with different pulse widths indicating different analog values. Then the network would read the signals every now and then which would be fed into the neurons.

PWM is pulse width modulation. Given an analog value it is possible to send a digital signal with a certain frequency that has different pulse width according to this value. The higher the analog value, the higher the width of the pulse. This is illustrated in Fig 4.1.

Since it is possible to send analog values using PWM, there is no need to use parallel to serial conversion. There are after all only five wires coming from the sensor modules, and it makes sense to sacrifice some wires for speed and feed them directly to the Koala robot rather than doing more complex

design to get them there. To get the analog values, it is possible to use a microcontroller with built-in analog to digital converters and then convert this analog value to a PWM and send it to the Koala robot. The modules can contain several films, and the films can have varied length. This could be adjusted by electronic components or software on the microcontroller for each module.

4.2 How to pass the signal from PVDF to the Koala?

The signal produced by the film needs some sort of amplification. It then should be converted to digital so it could be used by the Koala robot. But first, let's see if amplification is really needed.

4.2.1 Amplifying PVDF signals

PVDF characteristics have been discussed before, and now is time to use them to get the "right" signal out of the film. Upon impact PVDF produces a signal that is similar to illustration in Fig 4.2. Generally, weak signals produced by PVDF films need amplification. This obviously depends on application and the kind of signal required in the end. In this project, it is desired to get proper signals for each bump. The frequency or the signature of the signal is not as important as the strength of the signal. So if it is possible to somehow get unique values for different ranges of impacts then these values could be used to judge about the applied pressure.

There are various amplification circuits suggested by the manufacturer and also experimented with by others. Most of these designs were interested in getting some signal processing done on the output of the film, or separating the low and high frequencies. For example the films have been used by Taehee

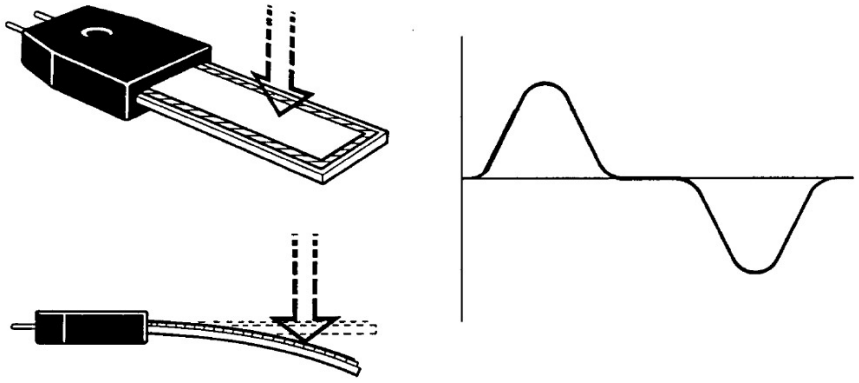

Figure 4.2: Deformation of the PVDF film is shown on the left. The signal on the right is created when the film is first pressed and then depressed. Picture taken from Kynar Technical Manual, Atochem Inc.

Kim [4] for tactile sensing. Kim has also done numerous experiments with the sensor, analysing the characteristics and frequency response of the sensor. For example, he discovered that noise cancellation of PVDF by means of analytical processing is not really effective. He had to use another PVDF film only for picking up noise, such that he could subtract this from the main signal in real time, to get the desired signal. This and similar techniques are used to get rid of the unwanted effects of the environment on film. Some of the possible amplification circuits are illustrated in Fig 4.3.

However for the purposes of this project, we do not need to have sophisticated circuits for analysing such characteristics. It is only desired to get unique values for unique impacts, so they can be distinguished from each other. For this reason the design was simplified to very few components. Since there is a need to convert the analog signal to some sort of digital, use of a microcontroller was considered. Microcontrollers are cheap nowadays and have the flexibility that is required in experimental design. Since the amount of pressure applied to the film would not be known exactly until the

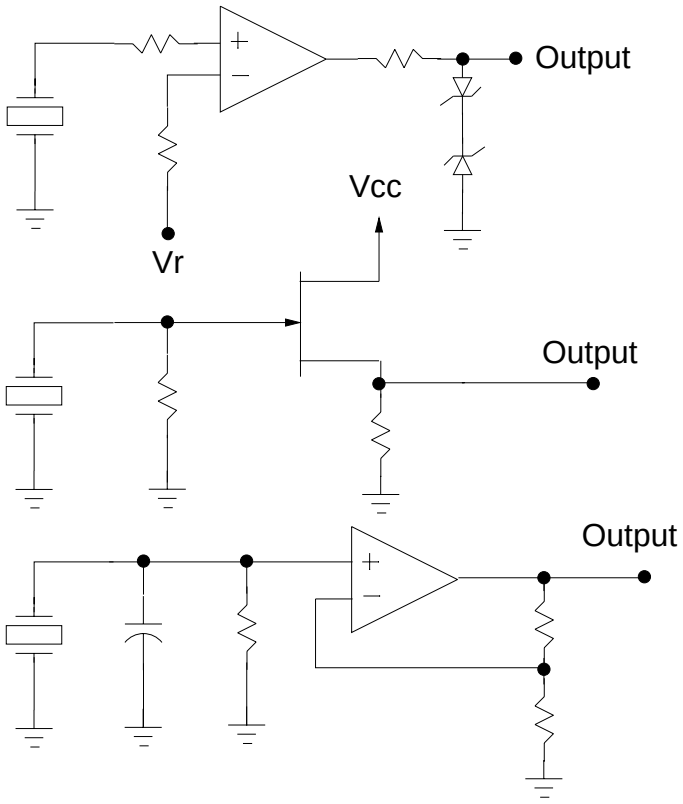

Figure 4.3: Several designs for the pre-amplification circuit [7],[4]. These designs were eventually discarded since there was no need for them.

mechanical design is finished and proper sponge material chosen, it is always a good idea to have some software flexibility to accommodate for this. The modules around the robot are all different sizes, but we wanted to use the same circuit for all of them to cut the costs, and a microcontroller is again advantageous.

The microcontrollers can have built-in analog inputs. Using these pins, it is possible to directly connect the PVDF and some simple components, like resistors, to get a certain signal upon impact. Basically the high impedance and leak current of the microcontroller pin would help to generate a voltage that could be read by the built-in analog to digital converter. This way, there is no need for extra amplification circuit, which would make the design

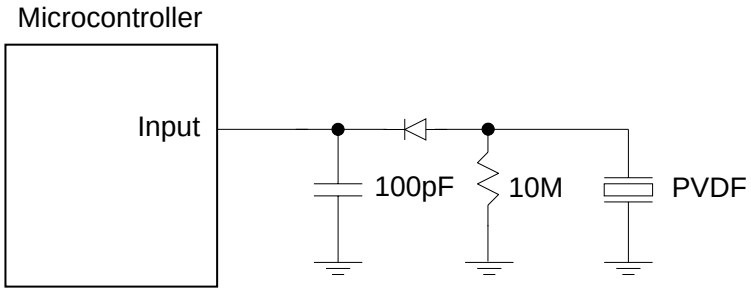

Figure 4.4: A partial circuit diagram that shows how PVDF is connected to the microcontroller.

cheaper and also makes the PCB smaller.

Each of the films has a resistor in parallel. The values of these depend on the size of the film. By varying the value the sensitivity of the film can be adjusted. This was done experimentally and a value around 10MΩ was found to be suitable for the short film and around 20MΩ for the long film.

The signal is then fed to a diode to be trimmed of the negative part of signal which is created by PVDF in each impact as described before. The negative part could be simply discarded since it is only an effect of the PVDF film when it is retracting to the original shape. Also a capacitor (100pF) was helpful in stabilising the signal before being sampled by the A/D. Because of the presence of the capacitor, the PVDF signal would decay, but this is not a problem since the idea is to get the relative information about the level of the signal and the precise amount of the voltage created by the impact is not really important. As long as higher impact creates a higher voltage comparing to a lower impact the system would work fine. Fig 4.4 shows the circuit diagram for connecting PVDF to the microcontroller.

4.2.2 Sending information to the Koala

The choice of the microcontroller is an issue to consider. It has to be cheap since there are going to be five of them used. It also should have A/D

channels for converting the analog signals to digital. It is also preferred to be small since there is not much space behind the plates to mount big PCBs. PIC12C671 was found to be a good choice. This microcontroller from Microchip comes in an 8-pin package. It has 6 I/O pins and 4 channels of A/D shared with digital I/Os. The size of EPROM[1] is 1K which is more than needed. The size of the RAM is 128 bytes. See [10] for more information on this microcontroller.

Each module then would have one microcontroller, and several films can attach to it. There are basically two ways to do this. First, one might attach each film to individual channels of the microcontroller and then, after processing them, send a value representing the state of impact. This is called multi-configuration mode. The second alternative is to connect all the piezo films together in series and connect this signal into one of the input channels. This is called single mode. This way we can use the number of films activated as an indication of how strong the bump is. The more films that deform, the higher the signal, and this could be sampled and used. Both approaches are interesting, so the design was modified such that it could be used either way by only changing some jumpers on PCBs. This also demands more complicated software to accommodate for it.

Each module would have a single output and also needs two wires for 5V supply and GND. To connect the modules in a tidy way, a ribbon cable with 7 wires was used that goes from one module to the next. Two of the wires are supply and each PCB would have a unique connection between one of the remaining five wires and the output of that particular module. This ribbon cable then goes to the Koala digital inputs. This is illustrated in Fig 4.5. For an illustration of the Koala I/Os see Appendix A. For more information on input-outputs of the Koala and other reference material see [9].

[1] Electronically Programmable Read Only Memory

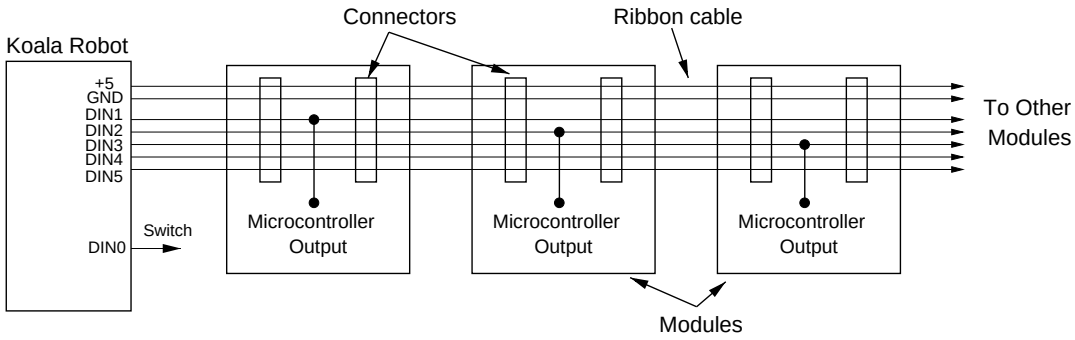

Figure 4.5: Block diagram of the modules connected with ribbon cable to the Koala robot. The connections are as follows: DIN1 to front-left, DIN2 to front-right, DIN3 to left-side, DIN4 to right-side and DIN5 to the back. DIN0 is connected to a switch for starting/stopping the software on robot.

4.3 PCB design

4.3.1 I/O pins

The microcontroller has 6 I/O pins. Three of them are used for the analog inputs. One is used to initialise the microcontroller for a particular module or configuration. Another I/O pin is used for two purposes: if this pin is high at power-on, it puts the microcontroller in test-mode which is used during development of microprocessor's software. The other purpose is to connect it to ground in normal operating mode. This will be described shortly. The last I/O pin is used as an output to send the signal to the robot. The block diagram of the PCB is illustrated in Fig 4.6 and the full circuit diagram is in Appendix B.

On board the microcontroller, there is only one A/D converter. So in order to read the analog inputs, the microcontroller has to switch between channels to read them one at a time. During the experiments it was found that connecting PVDF films with very high resistor values across them would put the analog input pins out of their specification. This caused major prob-

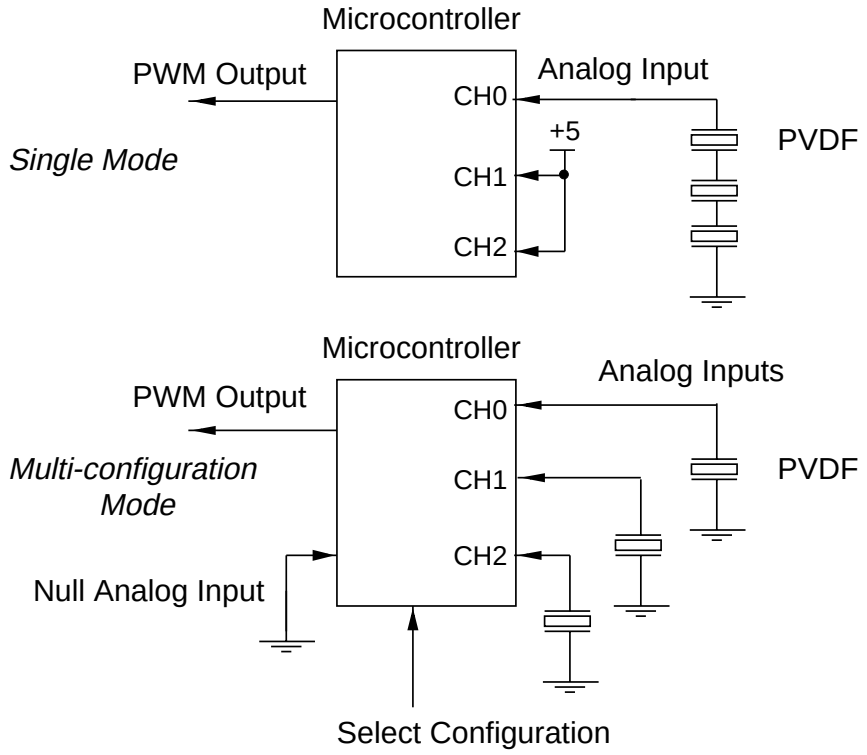

Figure 4.6: Block diagram of the electronics for each module. Null Analog and Select Configuration are not used in single mode.

lems when the microcontroller was in multi-configuration mode reading all the three channels. What happened was that when the A/D sampled one channel and switched to the next it was still reading the same value as before, even if a totally different signal was available in the new channel. The reason was that the internal sampling capacitor was not discharged during the time between switching the channels. This amount of time is recommended by the manufacturer to be around 10μs when the input pin is in the range of specification. This means that the external impedance should not be more than 10kΩ.

The resistor used by the circuit was far greater than this, so the internal capacitor did not have enough time to discharge and hold the large values for quite some time. To solve this problem, one of the analog channels was

connected to ground permanently. Now, every time a channel is read, this NULL channel is read afterwards to clear the capacitor, so the next channel can be read without delay.

4.3.2 Getting three states out of one digital input

The I/O pin used for selecting between different configurations is GP3 on the microcontroller. As described later, it should be possible to set the modules in one of the three configurations available. But there is only one I/O available to do this. So, the idea was to get three states out of this single input to solve the problem. This is achieved by exploiting the weak pull up available on the microcontroller.

As illustrated in Fig 4.7, the input pin is internally connected to 5V by a switch and a resistor. If the input is connected to 5V or GND, switching the weak pull-up to on or off would not have any effect. However, if the input is connected to ground through a resistor with the "right" value, then if weak pull-up is switched on the resistor would have a voltage across it because of the leak current of the pin and if this voltage gets bigger than the threshold (around 3V) the pin would read as one. If the weak pull-up is off, there would be no current and it would read as zero. So by switching the weak pull-up in software it is possible to get three states out of one input. The algorithm for this is as follows:

```
Turn weak pull up OFF
If Input=HIGH Then
   State = 1
Else
   Begin
   Turn weak pull up ON
   If Input=HIGH Then
      State = 2
   Else
      State = 0
   End
```

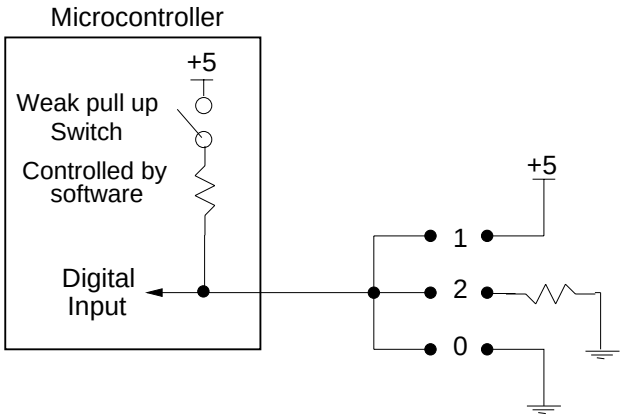

Figure 4.7: Exploiting weak pull-up to get three states out of one digital input.

4.3.3 Jumpers

The modules can not be exactly the same, since each of them can use different number of films and sizes. For this reason the PCB was designed in a way that it could be used for any of the modules by changing the resistor values appropriately. However, the software, described in the next chapter, is the same for all of them and would detect where it is used by using one of the inputs, GP3. This way modifying the software and making experiments was much easier. If there was a need to change a layout of the films for one of the modules, that module could be swapped with the current one, but the software could stay the same. Only some jumpers have to be altered to indicate the change.

There are two main modes for reading the analog signals. They can be chosen by using the jumpers on each module. However each module is independent of the others, so one could have some modules set to single mode and some others to multi-configuration mode. The software on the Koala should know this information, and then use the values received from each module accordingly.

MODE	J1	J2	J5	J6	J7	J8	J9
Single Mode	ON	ON	ON	OFF	OFF	OFF	OFF
Multi-configuration mode	OFF	OFF	OFF	ON	ON	ON	ON

Table 4.1: Jumper settings to choose between single mode and multi-configuration mode. J6 and J7 are ON if the jumper is in and are OFF if the jumper is not in.

The specification for jumper settings are illustrated in Table 4.1. J1 and J2 are used to tell the microcontroller about the mode, and J5-J9 are for re-wiring the lines such that the films connect to a single channel or individually to each channel. Jumper J3 is used for selecting between different configurations in multi-configuration mode. These are described in the next chapter. Jumper J4 should be OFF in normal use. However, if it is ON only on power up then the microcontroller will go to test-mode which then generates more complex output signals for different impacts for diagnosis purposes. Fig B.2 shows the layout of the PCB.

4.3.4 PCB connections

There are basically two kinds of connections on the PCB. One is the ribbon cable which is connected to CON1A and CON1B. The digital output of microcontroller is then connected to a unique wire of this ribbon cable for each module. The other connections are for PVDF films. Depending on the configuration used, they can be connected to CON2A, CON2B and CON2C. Each of these channels can have up to two PVDF films connected. For example, it is possible to connect one PVDF film to pins 1 and 2 of CON2A and another to pins 3 and 4. However, if there is need for only one PVDF film, pins 3 and 4 can be shorted by using a jumper. Obviously the number of resistors used should be changed accordingly too. It is important to notice

that films have polarity and they should be connected the right way.

The PCBs are connected to the back of the plates with some plastic railing glued to the plate. This was done to avoid making holes or putting screws in the middle of the plate which could interfere with the working of the PVDF film.

4.4 Summary

In this chapter the design of the PCB and circuit diagram was explained. However, details of using various inputs for different purposes are left to be described in detail in the next chapter. Details of modes and different configurations are also explained in the next chapter. The picture of the robot with sensor modules and PCBs connected around the robot is shown in Fig 4.8.

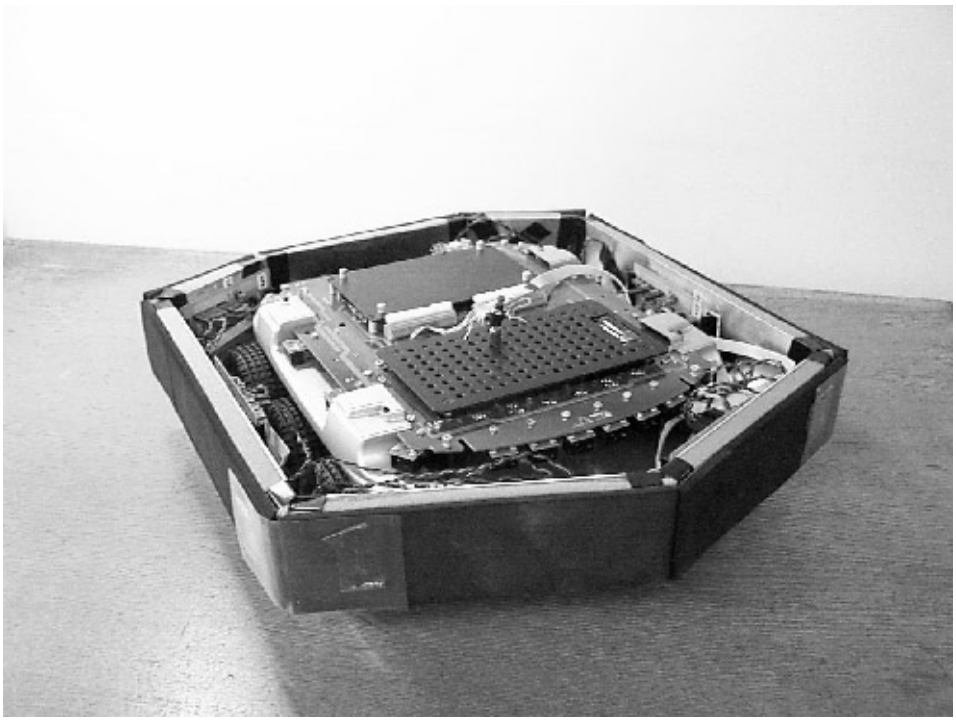

Figure 4.8: The Koala robot with final sensor modules and PCBs attached.

Chapter 5

Software Design On The Microcontroller

The software on the microcontroller is basically in two parts. The first part samples the analog signal and discards any noise. The second part sends the processed signal value to the software on the Koala robot through one of the digital outputs.

5.1 Modes

There are two main modes for reading the signal as described before.

- single mode. The signals are connected to one single input and the result to send to the Koala is a single number representing the amount of pressure on impact. This is an 8-bit number which is sent to the Koala using the PWM described earlier. See Fig 4.1.
- multi-configuration mode. The second mode is to connect each of films to a single channel. There could be up to three films connected. These signals are read in sequence, and converted to digital one at a time. Once a proper signal is detected, a value (8-bit) representing the state

of the impact is sent to the Koala. The major difference in this mode is that the signals received are processed and then a state is calculated accordingly. As an example, a state could be `RIGHT_HIGH`, which means that there has been an impact on the right side of that module and the impact is high in the amount of pressure applied. All of these are relative to that particular module, for example the front-left module. This way the Koala software knows more information about the state of impact for that module compared to single mode where only an analog value is sent representing the pressure of impact on all the films. Also, the microcontroller can do some pre-processing on the signal rather than just passing it to be processed later by the Koala.

In order to set the mode, jumpers J1 and J2 are used. If J1 and J2 are ON when the robot is turned on, the microcontroller goes into single mode. If they are both OFF, it goes to multi- configuration mode where there are three configurations as described in Table 5.1. Each configuration is designed for a particular side of the Koala according to different sizes and numbers of the films used, since each side is different from the other. For the front, it is possible to choose between two configurations. The effect of these layouts are tested by experiment and are described in later sections.

It is important to notice that these jumpers are read only on power on, which happens when the robot is switched on. So if the jumper settings are changed, the robot has to be switched off and on again (reseting the robot alone is not enough since it does not reset the microcontroller). The reason is that microcontroller inputs are limited in number and are overloaded with different functions, so are not constantly observed.

J3	Pins	Configuration
0	5,6	Front Bumper Config 1
1	1,2	Front Bumper Config 2 / Side Bumper
2	3,4	Back Bumper

Table 5.1: Jumper settings for different configurations of multi-configuration mode. Pins represents the location on PCB where the jumper has to be inserted.

5.2 Reading analog signals

The analog signals are first thresholded to discard any low level noise. The sources of noise are quite numerous as explained before. In single mode, the microcontroller continuously samples the input until it reads a signal which is more than a certain threshold. This will take it to the second stage where the signal is actually read. The signal is then sampled as fast as possible for approximately 4000 times in 200 ms. This interval was determined from experiment. In testing I found that most bumps into obstacles generate a cone shape pulse which is usually no longer than 200 ms and has a peak within the first 100 ms. The microcontroller records the reading as it goes along and tries to find the peak of this input. This peak is then sent to the Koala using PWM.

In multi-configuration mode, the channels are sampled one at a time, and the cycle repeats until one of them reads more than a certain threshold. The second stage is then to read all the channels, again one at a time, for 200 ms and record the maximum for each channel. After this is a third stage where these numbers are fed into a decision tree to decide between possible states. The decision trees depend on the configurations and layouts used and each one are unique. Then this decision or state, or the number which represents it, is sent to the Koala to be processed for the desired behaviour.

The channels are first compared to certain thresholds to decide about their levels. But each configuration can have a different set of thresholds. The analog values read in a certain module may be different from other modules, because each module has a different set of PVDF films. These threshold values are highly experimental and are documented in the code. As an example, the levels could be as follows:

$$\begin{aligned} \text{Analog value} > 150 &\implies \text{The signal is } \texttt{High} \\ 150 > \text{Analog value} > 20 &\implies \text{The signal is } \texttt{Low} \\ 20 > \text{Analog value} &\implies \text{The signal is } \texttt{Nil} \end{aligned}$$

The decision trees are basically common sense. If there is a film on the right and another on the left, and the channels read that the right one has detected a `High` impact and the left one has detected a `Low` impact, then by common sense we can say there is something strong on the right. All the decision trees are based on this concept. All of them are documented in the microcontroller's code, but for the sake of clarity one of them is reproduced here as an example.

The output states to be sent to the Koala are those described in Table 5.2. Each of these has a corresponding unique value such that the software in the Koala can decode it. The space of 8-bits is divided such that there are sufficient gaps to make the communications error free.

`ALL_NIL` is only used in the decision trees, and when sent to next stage for generating the PWM, is simply discarded, as there has not been a valid impact.

The decision tree for the Front Config 1 is as follows. Refer to Fig 5.1 for the layout of this configuration and Fig 5.2 for a graph of the decision tree.

States	Value	State	Value
ALL_HIGH	16	ALL_LOW	48
LEFT_HIGH	80	LEFT_LOW	112
RIGHT_HIGH	144	RIGHT_LOW	176
MIDDLE_HIGH	208	MIDDLE_LOW	240
ALL_NIL	0		

Table 5.2: Possible states and their corresponding values to send to the Koala in multi-configuration mode. Each state can have ±16 error.

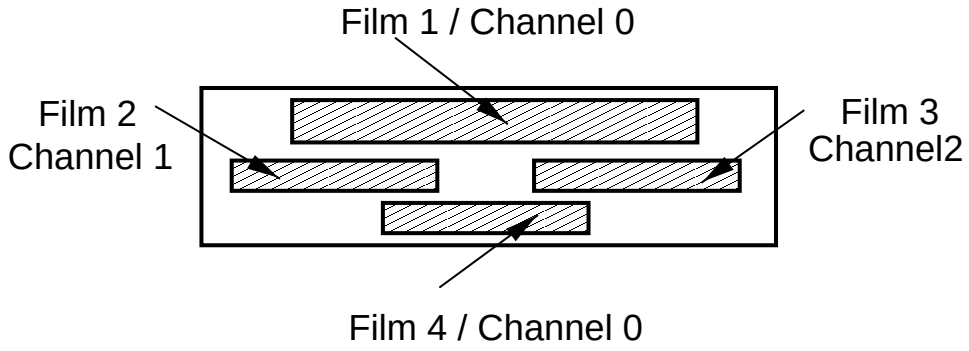

Figure 5.1: Layout of Front Config 1. Film one and 4 are connected together to channel 0. (Back view of the module)

```
If Channel 1 is High then
  If Channel 2 is High then
       Output is ALL_HIGH
  Else If Channel 2 is Low then
       Output = LEFT_LOW
  Else If Channel 2 is Nil then
       Output = LEFT_HIGH
Else If Channel 1 is Low then
  If Channel 2 is High then
       Output = RIGHT_LOW
  Else if Channel 2 is Low then
       Output = ALL_LOW
  Else if Channel 2 is Nil then
          If Channel 0 is High then
             Output =  MIDDLE_HIGH
          Else If Channel 0 is Low or Nil then
             Output = LEFT_LOW
Else If Channel 1 is Nil then
  If channel 2 is High
       Output = RIGHT_HIGH
  Else If Channel 2 is Low then
          If Channel 0 is High then
             Output = MIDDLE_HIGH
          Else If Channel 0 is Low or Nil then
             Output = RIGHT_LOW
  Else If Channel 2 is Nil then
          If Channel 0 is High then
             Output = MIDDLE_HIGH
          Else If Channel 0 is Low then
             Output = MIDDLE_LOW
          Else If Channel 0 is Nil then
             Output = ALL_NIL
```

This pre-processing will relieve the software on the Koala of some of the processing as well. It is possible to make more states, rather than having only three, but this would make the decision tree much more complex and it is also difficult to imagine if there would be much change in the behaviour that it is derived from it.

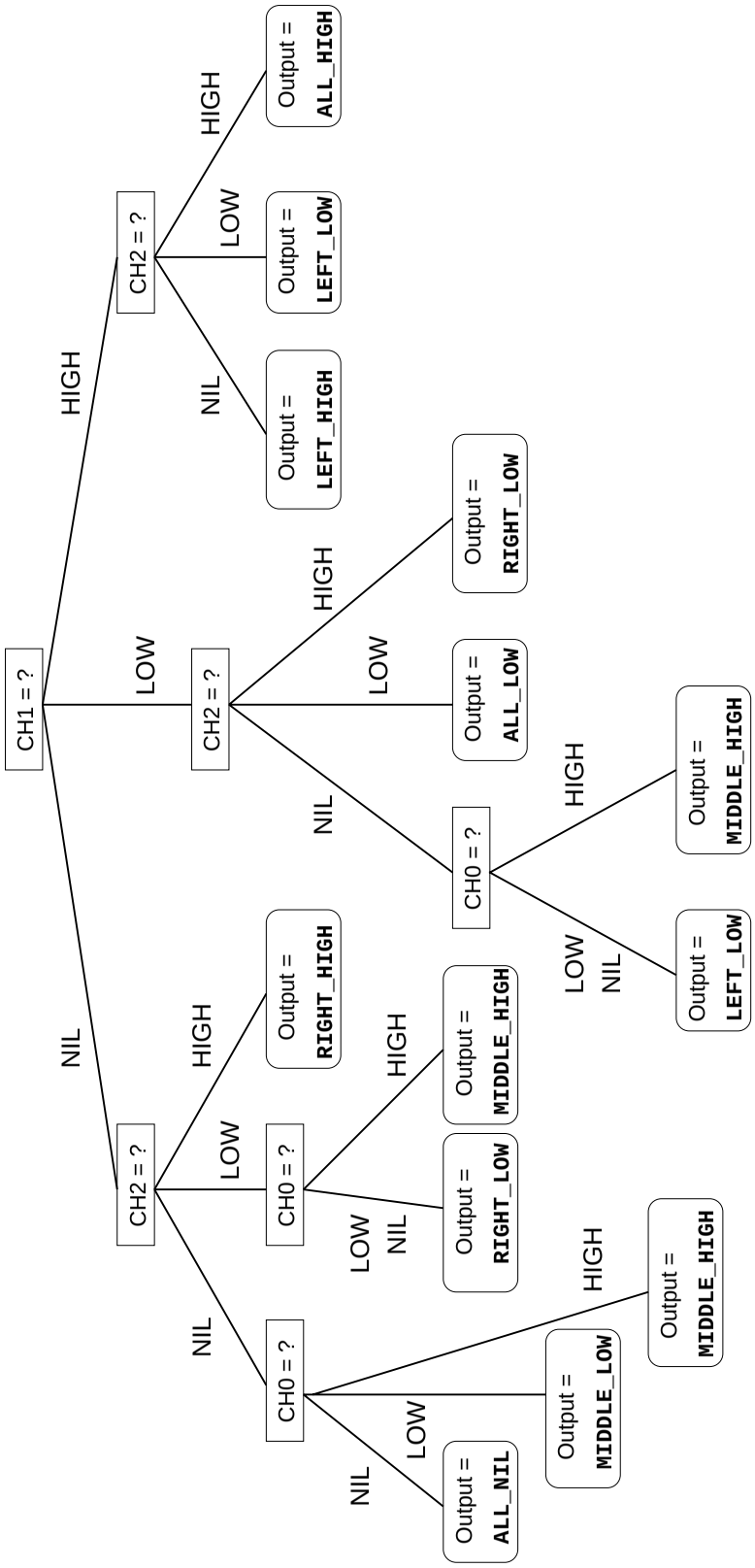

Figure 5.2: A graph of the decision tree implemented in microcontroller for Front Config 1. Lines represent different choices for each channel.

5.3 Sending PWM to the Koala robot

There are many ways to send the digital signal to the Koala. One is to send the value serially as an 8-bit number. This could be done synchronously using a clock or asynchronously with start and stop bits. Sometimes, if there are enough output pins available it is possible to send the values in parallel. However, this is not an option in this project.

Another method is to use PWM (Pulse Width Modulation) and send a digital pulse with analog information buried in the duration of the pulse.This method was chosen, since it could be quite fast and also is suitable for the current neural network that has been developed for the Koala robot.

So, using this method, depending on the operation mode of the microcontroller, an 8-bit value is passed to the subroutine responsible for generating the output. As illustrated in Fig 4.1 this value represents the width of the pulse; the higher the value, the wider the pulse. The frequency of the pulse train is constant and is primarily a matter of experimentation, since it has to be fast enough and at the same time easily decodable on the Koala's side.

The processor on board the Koala is unfortunately not so fast. It was realised during experiments that in order to use time sharing facilities of the Koala robot and also be able to detect the PWM on time, the frequency of the input PWM should not be too fast. A reasonable cycle time was around 100ms and sending three pulses was appropriate. More pulses would slow down the microcontroller in getting back to read more bumps. Fewer pulses might cause difficulty for the Koala to read the PWM when it has time to do so. If the Koala misses the first pulse in decoding, it can always start for the next one. See Fig 5.3 for a typical output generated for an impact to the left of a module set in multi-configuration mode.

The PWM routines in the microcontroller software are devised manually since there is no special hardware available in this microcontroller for

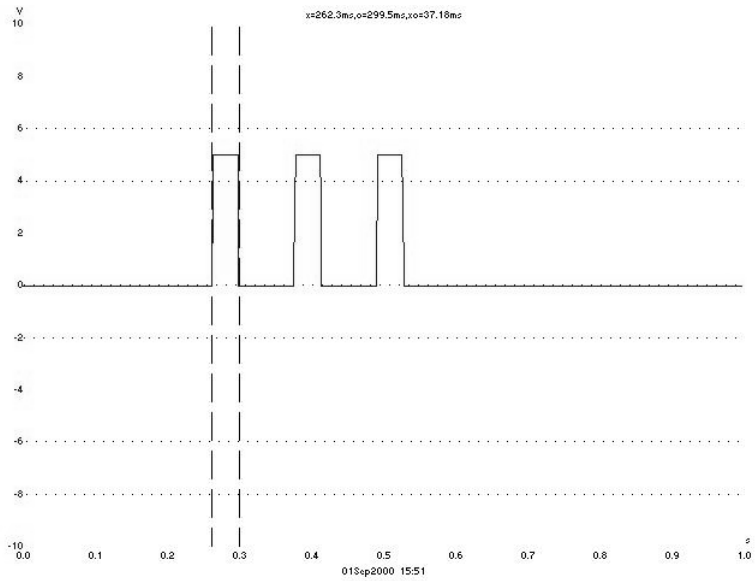

Figure 5.3: PWM pulse train generated for a typical impact in multi-configuration mode. Pulse width is 37 ms and the cycle time is 110 ms. This represents 83 out of 255 which is equivalent to `LEFT_HIGH`.

generating PWM. The microcontroller's timer module was used to generate various timings to create appropriate pulses on a digital output.

5.4 Summary

In this chapter the modes and configurations were discussed, and the method for sending information to the Koala was explained. Now the building blocks of the sensors are made. The time has come to perform a series of experiments to see what they are capable of. This is the subject of the next chapter.

Chapter 6

Characterisation Of The Sensors

In this chapter, a specification of the sensors is presented. A series of tests are done on all the modules to find the relationship between different kinds of bumps encountered and the signals returned by the sensors.

The idea behind the standard impact, explained in section 3.3, was used to test the modules when they were fully built. There are basically two series of tests done. One is to drop a certain weight from a certain height on to the modules (weight landing on top of the pads) and observe the signal received by the modules. The other test is to run the robot with various speeds into wall and observe the output of the modules.

6.1 Sphere dropping test

For this test the same sphere as in standard impact was used. Each module was placed horizontally and the sphere was dropped to different parts of the pad from a range of different heights. The sensor sent a signal to the Koala robot which was converted to get the final state. So everything is from the Koala's point of view which is what really matters. This specification is

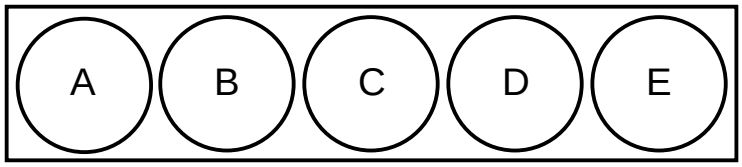

Figure 6.1: Areas tested in left and right front bumpers. Back view.

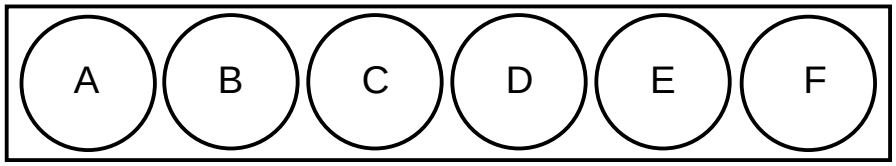

Figure 6.2: Areas tested in left and right side bumpers. Back view.

presented in the following tables.

6.1.1 Tables of results

For each module there is a corresponding area map. They represent the places that the impact between the sphere and the module happens. The idea is to observe the sensitivity of each area with respect to different amount of pressure applied.

All the modules are configured to be in multi configuration mode. Fig 6.1 represents the map of the areas tested for the front module. As usual this is viewed from the back of the module, so the left and right references can always be consistent with the moving direction of the robot. Table 6.1 and Table 6.2 show the output for each area for particular dropping height for front-right and front-left modules respectively. The dash in the tables means that no input was received. The height in all the tables is in centimetres.

For the side modules, the area map is presented in Fig 6.2 and the specification is in Table 6.3 and Table 6.4 for right-side and left-side modules respectively. The back area map is presented in Fig 6.3 and the results are in Table 6.5 and Table 6.6.

Height	A	B	C	D	E
2	-	-	-	-	-
3	-	-	MIDDLE_LOW	RIGHT_LOW	-
5	LEFT_LOW	LEFT_LOW	MIDDLE_LOW	RIGHT_LOW	-
7	LEFT_LOW	LEFT_LOW	MIDDLE_LOW	RIGHT_LOW	-
9	LEFT_LOW	LEFT_LOW	MIDDLE_HIGH	MIDDLE_LOW	RIGHT_LOW
10.5	LEFT_LOW	LEFT_LOW	MIDDLE_HIGH	MIDDLE_LOW	RIGHT_LOW
12.5	LEFT_LOW	LEFT_LOW	MIDDLE_HIGH	MIDDLE_LOW	RIGHT_LOW

Table 6.1: Results for front right module. A to E are illustrated in area map.

Height	A	B	C	D	E
2	-	-	-	-	-
3	-	LEFT_LOW	-	-	-
5	LEFT_LOW	LEFT_LOW	MIDDLE_LOW	MIDDLE_LOW	-
7	LEFT_LOW	LEFT_LOW	MIDDLE_LOW	MIDDLE_LOW	MIDDLE_LOW
9	LEFT_LOW	LEFT_LOW	MIDDLE_LOW	MIDDLE_LOW	MIDDLE_LOW
10.5	LEFT_LOW	LEFT_LOW	MIDDLE_LOW	MIDDLE_LOW	MIDDLE_LOW
12.5	LEFT_LOW	MIDDLE_LOW	MIDDLE_HIGH	MIDDLE_LOW	MIDDLE_LOW

Table 6.2: Results for front left module. A to E are illustrated in area map.

Height	A	B	C	D	E	F
4	-	-	-	-	-	-
5	-	-	ALL_LOW	ALL_LOW	RIGHT_LOW	-
7	LEFT_LOW	LEFT_LOW	LEFT_LOW	RIGHT_LOW	RIGHT_HIGH	-
9	LEFT_LOW	LEFT_LOW	LEFT_LOW	RIGHT_LOW	RIGHT_HIGH	RIGHT_LOW
10.5	LEFT_LOW	LEFT_LOW	LEFT_LOW	RIGHT_LOW	RIGHT_HIGH	RIGHT_LOW
12.5	LEFT_HIGH	LEFT_HIGH	LEFT_HIGH	RIGHT_HIGH	RIGHT_HIGH	RIGHT_HIGH

Table 6.3: Results for right side module. A to F are illustrated in area map.

Height	A	B	C	D	E	F
4	-	-	-	-	-	-
5	LEFT_LOW	LEFT_LOW	-	RIGHT_LOW	RIGHT_LOW	-
7	LEFT_LOW	LEFT_LOW	-	RIGHT_LOW	RIGHT_LOW	RIGHT_LOW
9	LEFT_LOW	LEFT_LOW	LEFT_LOW	RIGHT_LOW	RIGHT_LOW	RIGHT_LOW
10.5	LEFT_LOW	LEFT_LOW	LEFT_LOW	RIGHT_LOW	RIGHT_LOW	RIGHT_LOW
12.5	LEFT_LOW	LEFT_HIGH	LEFT_LOW	RIGHT_HIGH	RIGHT_HIGH	RIGHT_HIGH
14.5	LEFT_HIGH	LEFT_HIGH	LEFT_HIGH	RIGHT_HIGH	RIGHT_HIGH	RIGHT_HIGH

Table 6.4: Results for left side module. A to F are illustrated in area map.

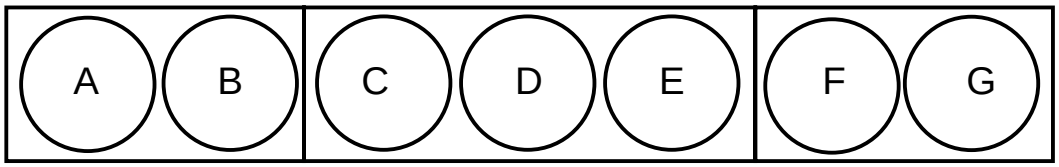

Figure 6.3: Areas tested in back bumper. Front view of the module.

Height	A	B	C	D
4	-	-	-	-
5	-	-	-	-
7	LEFT_LOW	LEFT_LOW	MIDDLE_LOW	-
9	LEFT_LOW	LEFT_LOW	MIDDLE_LOW	MIDDLE_LOW
10 - 18	LEFT_LOW	LEFT_LOW	MIDDLE_LOW	MIDDLE_LOW
19	LEFT_HIGH	LEFT_HIGH	MIDDLE_LOW	MIDDLE_HIGH

Table 6.5: Results for back module. A to D are illustrated in area map.

Height	E	F	G
4	-	-	-
5	-	-	RIGHT_LOW
7	MIDDLE_LOW	RIGHT_LOW	RIGHT_LOW
9	MIDDLE_LOW	RIGHT_LOW	RIGHT_LOW
10 - 18	MIDDLE_LOW	RIGHT_LOW	RIGHT_LOW
19	MIDDLE_LOW	RIGHT_HIGH	RIGHT_HIGH

Table 6.6: Results for back module. E to G are illustrated in area map.

Height of dropping (cm)	4	5	7	9	10.5	12.5	14.5	19
Energy on impact (J)	0.022	0.027	0.038	0.048	0.057	0.067	0.078	0.102

Table 6.7: Amount of energy on impact for each height used to drop the sphere.

For a better understanding of the amount of pressure applied on each impact the corresponding amount of energy for each height is presented in Table 6.7.

6.1.2 Evaluations

By looking at the tables, one can realise that the produced states are not always logical. Sometimes, when the sphere is dropped in the centre from a certain height, the sensor might send `LEFT_LOW` instead of `MIDDLE_LOW`. This is more pronounced when the height is so small that the amount of pressure is close to the limit of not being detected or when the height is so large that the entire module is affected by the vibration. This behaviour is the direct effect of the decision trees and the threshold values used in the microcontroller. Generally, microcontroller reads all the channels, for example three of them. The decision tree says that if the values read in different channels are very

close to each other, then `MIDDLE_LOW` or `ALL_LOW` is a good signal to send to the Koala. So, when an small impact happens, the difference between this channel and other channels might not be as much and so the signal would still be `MIDDLE_LOW`. Once the impact gets more and more strong, the decision tree would discriminate between the channels and would eventually send the right directional information.

Another reason for this can be the effect of the pads and the sponge. The pads usually transfer the impact to various areas of the module. So, even if the sphere is dropped in a certain area and is detected by one of the channels, the pressure might also be detected by other channels.

This unwanted effect can be reduced by adjusting the threshold values. As explained in the code, each module has a set of different threshold values for `NIL`, `LOW` and `HIGH` signals. By adjusting these threshold values, one can force the microcontroller to consider a certain bump as a low or a high impact.

As it can be seen in the tables, the best responses are usually around 10 to 12 cm as the height for dropping the sphere. The threshold values used to give this results, were obtained by experimenting in the environment were the robot was finally tested. The test is described in the next chapter.

6.2 Bumping into walls

The other series of tests done were to observe the state values sent by sensor modules for real impacts. By varying the speed of the robot, it is possible to get different results from sensor modules and to discover the lower and upper limits of detectable impacts. For these experiments the robot was ran into a flat stable wall, such as a wall of a room.

There are two tests. One is to run the robot forwards and the other is to run it backwards. The side bumpers were ignored for this test. For

the forward movement, there are five possible general angles as illustrated in Fig 6.4. The results are shown in Table 6.8. It is important to notice that the software on the Koala is capable of reading all the signals at once. So when the robot is running into the wall with high speed, it is possible that more than one module sends a signal to the Koala. In the table of results, the one that had a better meaning was listed. For example, if the robot is running into the wall and touching the wall with the corner of the front-right module, then front-right would send `RIGHT_LOW` and the right-side module would also send `LEFT_LOW` to the Koala. Since the robot is going forwards, the front-impact information is more useful, and this value is listed for the table of results.

The other test is to run the robot backwards. This could be done at three main angles as illustrated in Fig 6.5. The results are presented in Table 6.9. In this case, the impact can change the orientation of the robot. For example, when the Koala is going backwards with 45 degrees angle, upon impact it is still trying to push and this will change the orientation in such a way that the middle section of front module would eventually become parallel to the wall, which would then be detected by the microcontroller and would result in sending a different signal to the Koala. In practice the software should stop the robot if it detects the first impact. This way the sensor module would never detect the impact in the wrong section. However, since there

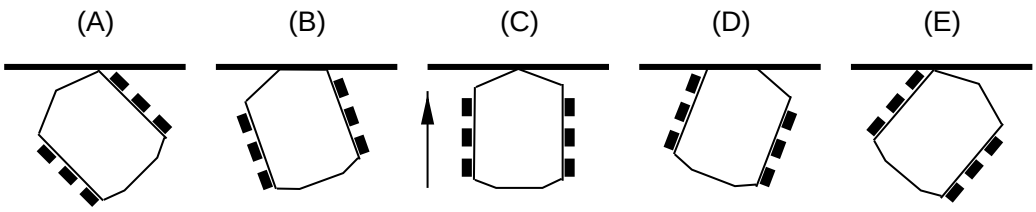

Figure 6.4: The robot is tested by going forward towards a wall with various angles.

V_s	V_r	A	B	C	D	E
10	3	R-SIDE	-	-	-	L-SIDE
		LEFT_LOW				RIGHT_LOW
20	6	R-SIDE	-	R-FRONT	-	L-SIDE
		LEFT_LOW		LEFT_LOW		RIGHT_LOW
30	9	R-SIDE	-	R-FRONT	L-FRONT	L-SIDE
		LEFT_LOW		LEFT_LOW	MIDDLE_LOW	RIGHT_LOW
40	12	R-SIDE	R-FRONT	R-FRONT	L-FRONT	L-SIDE
		LEFT_LOW	LEFT_LOW	LEFT_LOW	MIDDLE_LOW	RIGHT_LOW
70	21	R-FRONT	R-FRONT	R-FRONT	L-FRONT	L-FRONT
		MIDDLE_LOW	MIDDLE_LOW	LEFT_HIGH	LEFT_HIGH	LEFT_HIGH
100	30	R-FRONT	R-FRONT	R-FRONT	L-FRONT	L-FRONT
		RIGHT_LOW	LEFT_HIGH	LEFT_HIGH	MIDDLE_HIGH	LEFT_HIGH

Table 6.8: The results when the robot is going forward towards a wall. A-E are various angles. V_s is the direct value passed to the BIOS function to adjust the speed of the robot. V_r is the real speed of the robot in cm/s. R-something means signal form right something and L-something means signal form left something. At high speed the results can vary a lot due to high pressure applied on impact.

was no action performed in the testing software, this unwanted signal was detected and is listed in the table.

6.3 Summary

The specification presented in this chapter is based on the values that are sent to the Koala by the modules. The modules as explained in previous chapters interpret the PVDF signals and decide the states according to threshold values and decision trees. If any of these are changed, all the specification in the above tables become invalid. In fact, if a certain sensitivity is required,

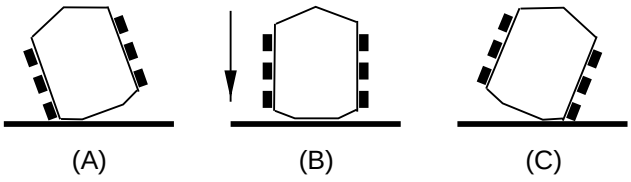

Figure 6.5: The robot is tested by going backwards to a wall with various angles.

V_s	V_r	A	B	C
10	3	-	-	-
20	6	MIDDLE_LOW	MIDDLE_LOW	MIDDLE_LOW
30	9	MIDDLE_LOW	MIDDLE_LOW	MIDDLE_LOW
40	12	LEFT_LOW	ALL_LOW	RIGHT_LOW
70	21	LEFT_LOW	MIDDLE_LOW	RIGHT_LOW
100	30	MIDDLE_LOW	MIDDLE_LOW	MIDDLE_LOW

Table 6.9: The results when the robot is going backwards to a wall. A-C are various angles. V_s is the direct value passed to the BIOS function to adjust the speed of the robot. V_r is the real speed of the robot in cm/s.

the best and easiest way to achieve it is by changing the various threshold values available in microcontroller's software. Each module has corresponding threshold values and if they were decreased, the module would generally become more sensitive and if they were increased, it would become less sensitive. Modifying threshold values is best done by experiment.

The tables presented in this chapter provide an idea of how the sensors might respond to different impacts. These however need to be taken as average and under laboratory conditions. For example, the response of the sensors could be different if they are bumped into sharp edges or if something bumps into them with very high speed.

Now that the sensors are made, and we know what they are capable of, it is time to make use of them. A few simple examples to use the sensors are explained in the next chapter.

Chapter 7

Software Design On The Koala Robot

When the modules detect impacts, they send PWM signals to Koala, which should be detected in real time. Then, using these values, the Koala software has to decide the action to perform. In this chapter, the structure of the Koala software is first discussed. Later, reception of the PWM by Koala and a couple of simple programs to display these values and use them in a limited way are presented.

7.1 Structure of the Koala software

As explained in the reference manual of the Koala robot, it is possible to use this robot in various ways, using internal or external processors. The assembly and C language functions of BIOS are provided for the internal processor. If an external processor is used, it is possible to use special commands sent through RS-232 to communicate with the processor on the Koala to access system resources.

In this project, the internal processor is used to create a few examples that use the sensor modules. It is possible to use an external processor as well;

however, the performance would depend on the speed of communications.

The Koala software used for the following examples is in C, and is based on the BIOS provided by manufacturer [8]. These are already-defined functions that can be used to interact with system resources. The program runs in time-sharing mode, having multiple processes that can run for 5 ms at a time before being switched to the next process by the time-sharing module.

The PWM signals generated by sensor modules were originally made to be used with the already-developed neural networks on the Koala. However to demonstrate the capabilities of the sensors, the software on the Koala was made from scratch using conventional algorithms. Unfortunately the limited time of this project did not permit to use the sensors with the neural network.

The five modules are connected to the Koala through digital input pins as shown in Fig 4.5. To read the sensors, a separate process is used. This reads the PWM inputs and then updates global variables accordingly. Then other processes can use these values to perform an action, like controlling the motors. The timing for inputs are done using a BIOS function, such as `tim_get_ticcount()`, which uses the timer in TIC module of the processor to keep track of time in milliseconds. Since there are three pulses available in each input when information is sent from the sensor modules, the Koala can skip through one pulse if it does not have time to service it. For example, it could be busy servicing previous bumps or reading new information on a different input.

Other processes can use the global variables to perform a particular action. The total number of processes is limited to thirty two by the manufacturer, however there is a practical upper limit of three processes when the sensors are used. Since each process takes 5 ms of CPU time, it is desired to give the control back to the process which reads PWM inputs as soon as possible, so it does not end up with errors in reading the pulses. It was found in practice that using more than three processes would cause the Koala to

misread the input lines. There is always a null process executing, imposed by the manufacturer, and another one which reads the PWM signals. It is thus possible to have another process to implement the behaviour required.

This is rather limiting. One way to solve this is to reduce the amount of time spent in each process, for example instead of 5 ms spend only 1 ms in each process. This will give us five times more processes to use! However this is only possible if the processor is actually faster, otherwise a lot of CPU power would be lost in switching between processes rather than doing something useful. Unfortunately 5 ms interval is set by the manufacturer and there is no way to change it.

Another way is to slow down the PWM pulse rates. This way the Koala would still be able to read the inputs as long as they are within the right range of input states. However, this will slow down the response time of the sensor modules, which in turn will make the program less efficient. Slow response to sensor modules might result in the robot bumping into an obstacle and pushing it for sometime without detecting it, which is not an acceptable behaviour. So, there is a trade-off between having more processes and having faster response time.

This might not be a problem if an external processor is used. As long as the speed of communication is as good as the response time, then everything will run smoothly. The external processor could be a computer which is usually ten times faster than the speed of processor on board. Any sort of time-sharing used would not cause any problem. However, only the internal processor was used in this project.

7.2 Reading PWM

The process that reads the inputs has to be able to read all of the inputs at the same time. First, all the inputs are in standby state. The software on

the Koala would observe the inputs waiting for a high to come. Each input is observed one at a time, and if is detected to go from low to high, then the state of that input is changed to active. If an input is active, the software waits until it goes low again which would represent a complete pulse width. During this time, the timer is observed and the pulse width is calculated in milliseconds. Using the cycle time of the pulse train, this value is then converted to the right proportion to end up as an 8-bit number. The cycle time is programmed manually, since it would not change once a proper speed is found to be suitable. This makes the design of the software easier.

Now the global variables are updated with this new value, and the program continues to read new signals. If a signal is just read, there is a suspend time in the Koala software (around 1 second) before it can be activated again. This is done to prevent reporting the same bump over and over again.

If the sensor module is in single mode, the new global value can be used directly as the 8-bit value representing the amount of pressure applied to all films. However, if the sensor module is in multi-configuration mode then it needs to be adjusted to the closest value that would represent a valid state. There are 8 states in total as described in Table 5.2 which divide the 8-bit space equally. Once an 8-bit value is read, it is then converted using the following mathematical formula to get one of these 8 states. This is done to prevent loss or change of states due to bad communication between the Koala and the sensor module.

$$\begin{aligned} \text{State Value} &= 2 \times \text{HalfScale} \times round\Big(\frac{\text{AnalogValue} - \text{HalfScale}}{2 \times \text{HalfScale}}\Big) \\ &+ \text{HalfScale} \end{aligned} \tag{7.1}$$

Here *Half Scale* is the maximum error in either direction. Since the 8-bit number is divided into 8 regions, each region is 32 points wide. The

middle number would represent this region and so the maximum error is 16 in either direction. The state value then could be used in other processes to do a certain action or in a decision tree for avoiding bumps and so on. In the actual code, the division could be replaced by bit-shifting to make this calculation much faster.

In multi-configuration mode, if PWM signals are from Front Config 1 or Back, then relative direction would be interpreted as normal. However the situation is different for Front Config 2 or Side configurations. As mentioned before the right or left directions are viewed from the back of the modules. So the left of the right-side module would be towards the back, while the left of left-side module would be towards the front. In using the states for this configuration, this asymmetry in input states should be considered in implementing algorithms.

7.3 Obstacle avoidance in single mode

So far, the routines for getting the signal from modules to global variables have been discussed. It is now time to use these and see what can be achieved. In this and the following section two examples are presented for a basic obstacle avoidance algorithm that uses only the two front modules.

The first program uses the modules configured in single mode. Each module sends an analog value to the Koala which is basically the amount of pressure of signal received from all the sensors on that module. Using this value the algorithm tries to avoid obstacles detected in front and continues searching in other directions.

There are two algorithms done for this program. The first one uses the module in a binary fashion, i.e. it does not use the pressure information. The second algorithm uses pressure information to decide the actions to be taken.

7.3.1 Algorithm 1: Without pressure information

The first algorithm is as follows. To detect an impact, the pressure received by the Koala from a sensor module should be more than a certain threshold to be classified as an impact on that module. Once an impact is detected by the Koala, then the Koala turns toward opposite direction for a certain random angle. For example, if the front left module detected an obstacle and sent an analog value to the Koala, the robot would then turn to the right for a random angle. This angle has a range to select from, for example, turn to something between 70 and 150 degrees. Also before performing the turn, the Koala goes backwards a little, so it can clear the obstacle much more smoothly.

This program was tested in practice. The robot is capable of searching the environment. However due to random behaviour, it does not have a systematic way of searching. So if there are narrow corridors in the environment, the robot spends quite some time trying to get out of them.

To get better results, the other modules should also be used. For example the back sensor could be used to tell the robot if front and the back of the robot are both blocked and it would have a better chance of escaping, if it turns to a perpendicular direction. For obstacle avoidance in general it is good to use some other sensor or means of detecting the speed of the robot. For example, in the Koala, it is possible to monitor the speed of the motors. Using this it would be possible to detect if the robot is stuck somewhere and try to perform a particular "escape action" to get away.

7.3.2 Algorithm 2: With pressure information

The second algorithm uses pressure information. The basic structure of this algorithm is the same as the other algorithm except for the amount of turn to perform. To calculate the turn angle, it is possible to use the pressure

information. For example, if the Koala bumps into something very hard, then it could turn faster to get away quickly. On the other hand, if the signal was not as strong, then it could go slowly since this is probably a little touch to some permanent obstacle and the robot only needs to adjust the angle of movement a little.

This algorithm was also tested in practice. It was possible to observe that the robot would turn faster when it bumps into something with full speed, or when it was kicked! However, the different turn angles did not make a big difference in search capability of the robot. In this case the pressure information was acting as the random angle. When it bumped into something, depending on the sharpness of the object bumped, the turn angle was calculated. Since a random environment has a random number of sharp object in random locations, the overall result was pretty similar to the first algorithm.

7.4 Obstacle avoidance in multi-configuration mode

Obstacle avoidance can also be done when modules are set to multi-configuration mode. Since more directional information is available, it is possible to have a little bit more complicated routine to avoid obstacles.

Here, the angle for turning is not random anymore. As before, if the left module sends information indicating a bump, the robot turns to the opposite direction (right). The difference is in the angle which depends on the states sent from the sensor modules.

When the robot hits an obstacle such as wall with the left module, there are three basic possibilities. It could hit to the left of the module, which means it has to turn something like 90 degrees to the right to clear it. On the other hand, if it hits in the middle, the angle to turn should be a little

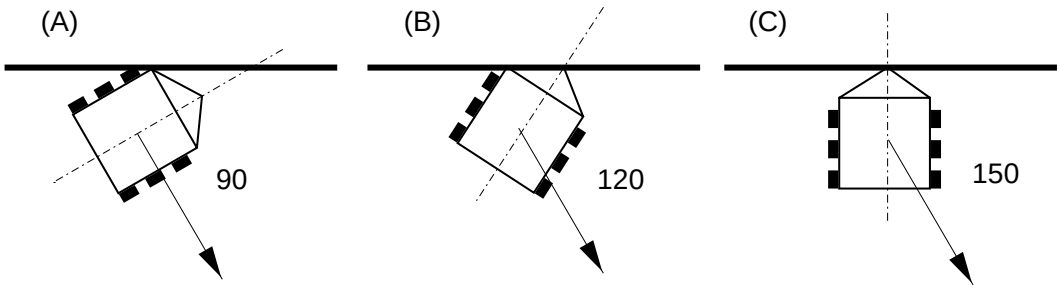

Figure 7.1: Three possible states when the robot hits a wall with the left module. The angles of turn are illustrated. (A) is hitting the left of left module, (B) is hitting the middle of left module and (C) is hitting the right of left module.

more, around 120 degrees. This is because the robot is going more straight into the wall comparing to the last condition. If it hits the right section of the left module, then it is going almost perpendicularly towards the obstacle, and it should turn around 150 degrees to get away. The reason behind the wide angles is to make the robot go and explore as much as possible. These conditions are illustrated in Fig 7.1.

On each impact, if the impact is high in the amount of pressure applied, the robot would go back a little before turning. Otherwise it will turn straight away without going backwards.

This program was also tested in practice. It was capable of searching in a better way comparing to the previous two algorithms. For example, it explored areas were the others struggled to go into or to get out of. However, it still was not perfect in exploring every corner and accessible area of the environment. The same argument as having more sensors would also apply to this algorithm.

7.5 Summary

In this chapter the basic structure of the Koala software was explained and a few algorithms and example code were given to illustrate how the sensor modules could be used in general. The examples were deliberately chosen to be simple, so the interaction between the sensor modules and the Koala would be easier to follow for someone who wants to know how to use the sensors for a particular application. For more interesting but simple algorithms to use see Braitenberg's book [11]. More complex algorithms will be presented in later chapters.

Chapter 8

Corridor Following Algorithm

In this chapter a more complex algorithm is presented to show how much the sensor modules are capable of. For this purpose, corridor following was chosen. This is a well studied subject and even if it is used in a limited way it would still be able to show the capabilities of the bumper sensors.

8.1 The task

The task is to run the robot in corridors with two kinds of junctions. One kind is the L-junction which is basically a 90 degrees bend to left or to right. The other, is T-junction. At a T-junction the robot has two choices: to go left or right. The corridor has straight flat walls and is twice as wide as the robot, approximately.

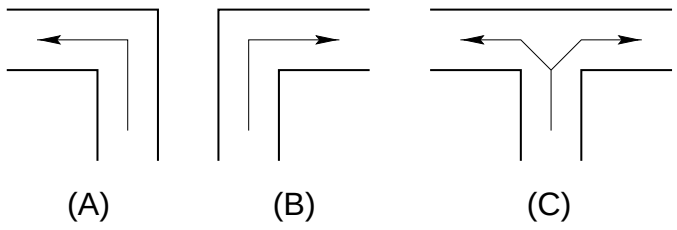

Figure 8.1: (A) L-junction to left. (B) L-junction to right. (C) T-junction.

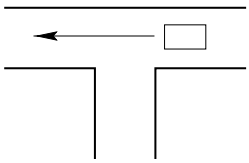

Figure 8.2: The robot will never be in this situation. This is done to simplify the task.

A combination of the corridors and the two kinds of junctions represents the environment which the robot has to interact with. To make everything simple only the bumper sensors developed in this project are used to guide the robot. However, in practice usually other sensors are used for this task since bumper sensors are not the ideal choice. For example, IR sensors are a better choice since the robot can get information from all around it without the need to interact actively with the environment all the time.

Another simplification is to avoid branches. For example, at a T-junction the robot always enters the junction from the branch. This means that the robot is never able to recognise a branch that goes away from the main corridor. This is illustrated in Fig 8.2.

The idea is to give the robot a series of instructions which the robot should follow in a maze. For example, the robot can have this instruction:

> At the first T-junction turn left and at the second T-junction turn right and at the next T-junction turn right and stop.

The robot should be able to distinguish between the L and T junctions. It should ignore the L-junctions since there are no choices available there. Once it detects a T-junction, then using the instructions it should be able to decide which way to turn and it should continue doing this until it reaches the end of the instructions. A possible environment is illustrated in Fig 8.3.

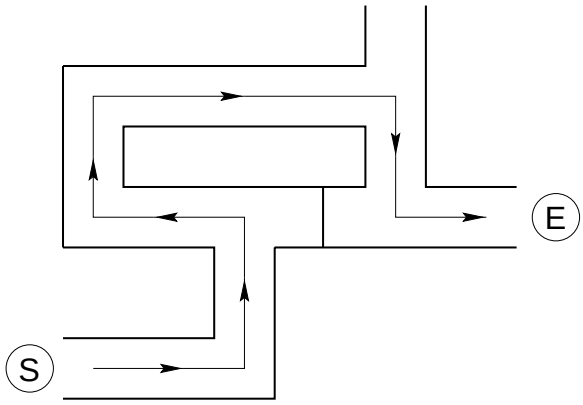

Figure 8.3: An example of an environment consisting of L and T junctions connected with corridors. As illustrated, a possible instruction is to follow these: `LEFT,RIGHT` *and then* `LEFT`*. S is starting point and E is the end point.*

8.2 The design

Before we go through the details of the algorithm, it is good to consider some guidelines. The first consideration is to see which mode the sensors should be in. For this task multi configuration mode is better since it gives much more directional information on impacts which is exactly what the robot needs. The robot would need to use all the sensors around it to make sure there are no blind spots. The software on the Koala should read all the inputs and use the timer and current states to decide for the next actions to make.

The instructions are given to the robot using an array that contains a series of values that could be `LEFT` or `RIGHT`. Once the algorithm reaches the end of this array, it would consider it as the end of the instructions and would then stop.

The algorithm is heavily based on timings and states. It starts with a predefined state, like standby, and every time that it receives state signals from modules it would update the internal state and continues this cycle.

First the robot should be capable of following a wall in a corridor. Since

there are no IR sensors, the robot could only recognise a wall by bumping into it. As usual it is always better to take advantage of the physical characteristics of the environment rather than programming it. During practice it was discovered that if the robot can simply slide along the walls without actually detecting it, it is good enough for the purpose. So a smooth material, like a thin sheet of plastic, was attached on top of the front corners. See Fig 3.4. This will not prevent detecting an impact but when the robot is going with an angle towards a wall, it would make the robot slide and basically follow the wall in parallel. This proved to be a very useful feature.

However, if the robot detects the wall with one of the corners of the front modules, then it must be going with a sharp angle towards a wall and it needs to adjust the direction of its movement, so it can become parallel to the wall. This means the Koala should rotate something around 30 to 45 degrees on average. The angle does not have to be precise since if the robot has a slight angle with the wall it will eventually get straight by sliding. This behaviour is illustrated in Fig 8.4. If the robot adjusts the angle in this way, it can then presume that it is following the left or the right wall of the corridor. If this information is saved in the internal state, it can be used later on when it comes to a junction as explained in the next section.

8.3 Detecting junctions

L and T junctions are very much the same. One main routine could be used and the result of that would tell the robot about the kind of the junction that is encountering. Using the internal state that tells the robot which wall is following, it can then use two different algorithms to decide about the junction, one algorithm for the right and another for the left side. The two algorithms are mirrors of each other. To make the algorithm easier to understand, it is described in English rather than using pseudo code. For

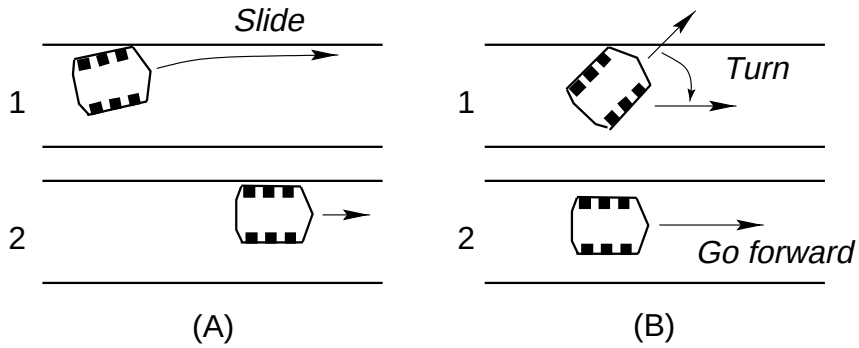

Figure 8.4: (A) In this situation the robot would adjust the angle by sliding. (B) This is an example when the robot goes towards a wall and detects it on the left corner, it then turns to right to adjust the angle.

example if the robot is following the right wall of the corridor, the algorithm to detect a junction is as follows.

8.3.1 The algorithm

The robot is going forwards and knows that it is following the right wall. If it hits something in the front, then it must be the wall of a junction. Now, it has to discover what kind of a junction it is. First, assuming that there is a wall on the right, it turns to the left with a 90 degrees turn. Then starts a timer and goes forward. If the timer runs out without bumping into anything then this path is clear. So this could be a L-junction to left. But it could also be a T-junction. To check for this, the robot would stop when the timer runs out and start going backwards. It uses another timer, and goes backwards until that timer runs out. If nothing was bumped into, then this must be a T-junction. Otherwise if it hits something on the back before the timer runs out, then the original decision was right and it was a L-junction to the left. At this point the robot should know exactly what to do. If it is a L-junction it has to go forwards, since it does not have any other choice. On the other hand, if it is a T-junction it has to lookup in the instruction array and select

the current instruction. If it says to go to left, it has to go forwards since it is pointing to left. If it is to right, it has to turn 180 degrees and go right.

What if, in the first 90 degrees turn to left, it runs into a wall before the timer runs out? This means that the left wall is blocked and the only way out should be to the right. So the robot can instantly realise that this must be a L-junction to the right. It should turn 180 degrees and goto right.

The same algorithm would apply if the robot is following the left wall, except that the turning and the interpretations should be mirrored.

8.3.2 Special cases

There are, however, a few points to consider. What if the robot "thinks" that is following the right wall but, in fact, it is following the left. This can happen quite easily. For example, the robot bumps into the right wall with 40 degrees angle and then adjusts the turn and updates the internal state about which wall is following. But the turn was a little too much and now, the more the robot goes forward, the closer it gets to the other side until it is following the left wall.

In this situation when the robot bumps into the front wall of a junction, it tries to turn into the wrong direction. But then it would instantly detect a wall since it can't even turn into it. It should then turn to opposite direction knowing that this can not be a T-junction. What this means is that the algorithm can handle the situation even if what it thinks is not the same as what is actually going on, and this is always a very useful feature to have.

The robot would then continue this cycle until all the instructions are performed and then it stops and goes round and round indicating that it has found the destination.

Another problem that can occur is when the robot is trying to turn, but it is touching a wall in parallel on the opposite side. For example when the robot starts turning to left, the back of the robot tries to push the wall on the

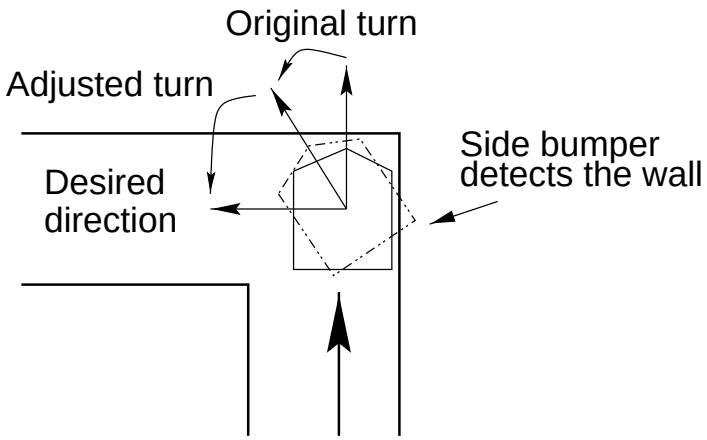

Figure 8.5: Adjusting the 90 degrees turn using the side bumper.

right but it can not do this. So the turn is affected and the robot would turn around 45 degrees instead of 90 degrees. This is where side bumpers become useful. When the robot starts turning, the corresponding side bumper (in this case right side) would detect a bump and the robot can then go forward to make some space between itself and the wall, and then turn some more to adjust the turn to the desired 90 degrees. This is illustrated in Fig 8.5.

8.4 Results

The algorithm was tested in a set of example environments. Fig 8.6 shows a portion of the maze. The Koala was capable of finding the destination as long as the environment was simple. For example, the walls are always flat and the robot starts almost parallel to the walls of the corridors. If the robot bumps into a corridor wall and recognises it as a junction rather than a wall, it then starts performing the algorithm for detecting the type of the junction where no junction exists, so it will never reach the destination. A single mistake is enough to make it fail, since everything depends on past actions and the errors accumulate.

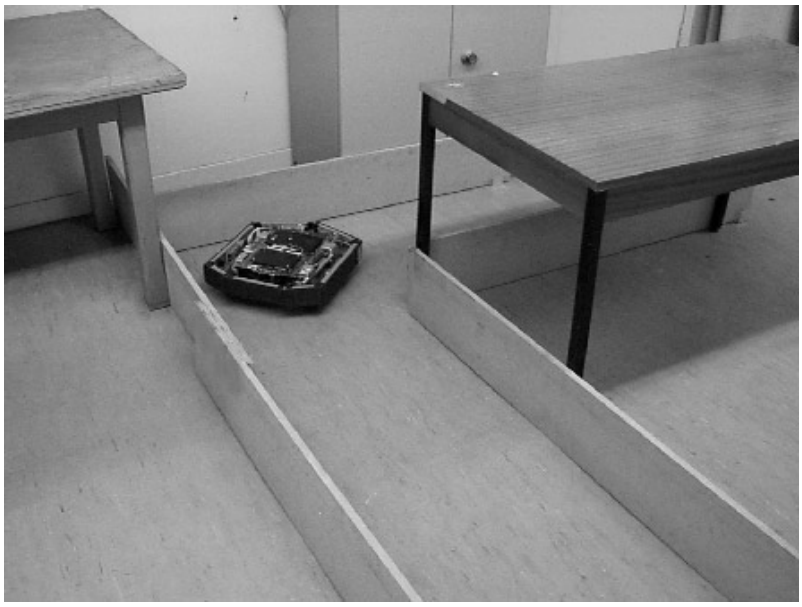

Figure 8.6: Picture of the robot during a test in a partial environment with corridors.

This however is not a serious problem, since the idea is to demonstrate the capabilities of the sensors. Eventually, in any realistic application, the bumper sensors should be used in conjunction with other sensors.

A possible path is presented in Fig 8.7. The path line represents the tracking of the position of the front of the robot where the two front modules come together.

8.5 Summary

The algorithm presented in this chapter is one of the many possible algorithms for corridor and maze following. As with every sensor, there are advantages and limitations in using this sensor for particular environments. What is deducible from the results, as is arguable in theory, is that the bump sensors on themselves are not enough in practical applications.

The general results and conclusion for the project are presented in the

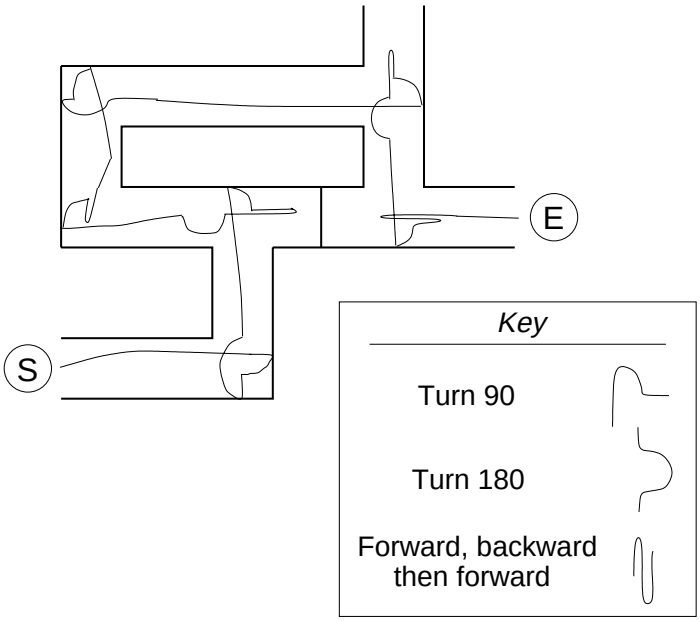

Figure 8.7: A possible path in an example environment. S is the starting point and E is the end point.

next chapter.

Chapter 9

Conclusion And Future Work

The project's conclusion is presented in this chapter, and possible ideas are discussed for future use of the sensors. As with any engineering project, the ultimate goal is to meet the desired specification and to test the product and see if it really meets various criteria considered in the project's requirements.

The goal of this project was to make pressure sensitive sensors that could detect bumps form all angles, without having blind spots. They should be robust and capable of working in different environments and give sensible information to the main program, on the Koala or elsewhere, to decide what action to make.

To see if the specification discussed in earlier chapters has been achieved, a relatively complex algorithm was developed to test the robot in following corridors with different kinds of junctions with given instructions and see if the robot is capable of performing the task. There are however two issues to consider which should not be confounded. One is the ability of the sensors to detect bumps from different angles, or calculate the amount of pressure applied, and pass it up to the main program. The other is the main program that is going to use these values and perform a particular algorithm. The robot under test relies on both of these to succeed in any task. However the judgement should consider the differences between the two. If a bad

algorithm is used and the behaviour of the robot is not satisfactory, this does not mean that the sensors are useless. On the other hand if the sensors don't pass consistent information to the algorithm, then however clever the algorithm, the overall task is doomed to fail.

The other point to consider is to see if the sensors are easy to use and do not make the algorithm very complex and time consuming to write. Using the sensor modules was quite easy in the algorithm that was developed in corridor following. The difficult part is perhaps the lack of processing power on the Koala robot. So if anything serious is going to be programmed, it would be pretty difficult since the Koala processor is not fast enough.

9.1 Evaluations

The sensors are generally good in detecting the bumps but they also have their limitations. The sensitivity of the sensors could be adjusted using the threshold values in the microcontroller's code. Using these threshold values, certain weak bumps could be ignored. If they are adjusted to be very sensitive then each bump might generate signals in many other modules even if they have not actually bumped into anything. So there is a balance to be achieved for different kinds of environments.

If the sensors are used in single mode the orientation information for each module is lost and only a value representing the amount of pressure is passed up to the Koala. In practice this mode proved to be of less use than multi-configuration mode since it was always more desirable to have directional information than the amount of pressure received from a module on each impact. Having a fuzzy concept of `HIGH` and `LOW` amount of pressure used in multi configuration mode was also very useful. The same applies to the areas on the pads, such as `LEFT`, `MIDDLE` and `RIGHT` which had fuzzy boundaries and it was always relative to the particular module. This way there was no

need for precise and costly calculations to end up with a precise value that might eventually not be used by the algorithm in the Koala since it did not need that much precision.

In general the amount of uncertainty in impacts is very high. When the robot bumps into something, the impact can change the orientation of the robot. The robot can not precisely calculate what the new orientation is even if it knows what the original approaching angle was. For example, if it bumped into a wall with 30 degrees, because of the strong push it might carry on pushing until one of the front modules becomes parallel to the wall. By now it has 45 degrees angle. But this might not happen every time. So there is always an uncertainty involved in decision making.

This problem was very pronounced in the corridor following program. If there was even a single bump into something unexpected, the whole routine would fail. Detecting that particular situation was then very difficult since it is difficult to guess when it would happen. In this experiment it became quite clear that even a simple task is difficult to accomplish using only the bumper sensors. They would be best if they are used as secondary precaution sensors helping other sensors to figure out what is going on in the environment.

For example a combination of IR sensors and bumper sensors are ideal for maze and corridor following. The IRs can detect the walls and if they fail to detect something due to unavoidable blind spots, then the bumper sensors would be able to say if there is something wrong when the robot runs into a wall without detecting it. The algorithm can then correct the action and continue with the task.

Another problem is when the robot bumps into a wall when the front module is in parallel with the front wall and the robot is going with such an angle that when the impact happens, the front module touches the wall with all the module's surface. When this happens, the front pad would be shifted back and this would press all the sensors into the soft sponge in the

back of the PVDF films. This way the films don't bend, so the impact is not detected. This happens very rarely but there are ways to solve it, if desired. One way is to make the modules curved. This way they would always bend on impact with a wall. The problem is that making curved modules would cost more and is mechanically more complex to implement. An alternative way is to attach bars or similar objects between the soft sponge and the vertical plate. This way, when the impact happens, the areas with the bar would prevent the film from coming back as much as other parts of the film which bends the film and eventually the impact is detected.

9.2 Future work and final thoughts

Due to limited amount of time available in this project there are still many possible experiments left to be done. Using the bumper sensors in conjunction with IRs as described in the last section is a good start. This will present the practical problems that might be encountered in a more realistic situation. After all even if there is a problem when bumper sensors are used by themselves, there is no point to solve it if this problem does not happen in realistic situations.

Another series of experiments left to do is changing the layouts used in modules. For the front module, there is a possibility to use two different configurations. This is already implemented in the software, but it should be tested in practice to see if different layouts would have a big difference on performance of the sensor modules. To test this, it is better to first develop a practical application and then switch the layouts to see if the performance would change using statistical techniques. The important point is that the test should be of practical use, since testing it in laboratory conditions and simplified environments would not necessarily give a valid conclusion.

Another idea is to use the sensors outdoors, which was the place they

are intended to be used, and observe the behaviour of the robot and the sensor modules. As in many robotics applications, many of the limitations of the sensors are discovered in practice and it is pretty difficult to guess what might go wrong beforehand. The usual way is to develop some program, run it and when the robot does something peculiar then start thinking how it can be avoided and the performance bettered.

9.3 Summary

To summarise, the conclusion is that the sensors have to be used in many practical applications in different environments, so they could be judged if they are useful for their task. The sensors were successfully developed in this project, but whether they are successful in practice is to be seen in the future when they have been fully used and tested.

References

[1] Barbara Webb, Using robots to model animals, a cricket test. Robotics & Autonomous Systems 16 (117-134), 1995

[2] R. R. Harrison, A robust VLSI motion sensor, Autonomous Robots 7, 1999, (211-224)

[3] Graeme Watson, An Active Antenna Sensor for the Cricket Robot, MSc dissertation, University of Edinburgh, 2000

[4] Taehee Kim, Development of PVDF tactile dynamic sensing in a behaviour-based assembly robot, PhD Thesis. University of Edinburgh 1996

[5] Tim Chapman, B. Webb, A neuromorphic hair sensor model of wind-mediated escape in the cricket, International Journal of Neural Systems, Vol 9, No. 5, October 1999, (397-403)

[6] B. Webb, T. Scutt, A simple latency dependent spiking neuron model of cricket phonotaxis, Biological Cybernetics 82(3):247-269, 2000

[7] Kynar^{TM} technical manual. Atochem Sensors Inc.

[8] Koala BIOS Manual, K-team.

[9] Koala Reference Manual, K-team.

[10] PIC12C67X reference Manual, Microchip Technology Inc.

[11] Valentino Braitenberg, Vehicles: Experiments in Synthetic Psychology, MIT Press, Cambridge Massachusetts, 1984 (ISBN 0-262-52112-1 paper)

[12] Wordbook encyclopedia, IBM corporation, 1998.

Appendix A

The Koala Robot

This appendix contains the picture of the Koala after the sensor modules have been mounted. Also, other specifications of the Koala robot, such as input/output connections are also represented. For further information about the Koala robot see [8] and [9] or online at www.k-team.com.

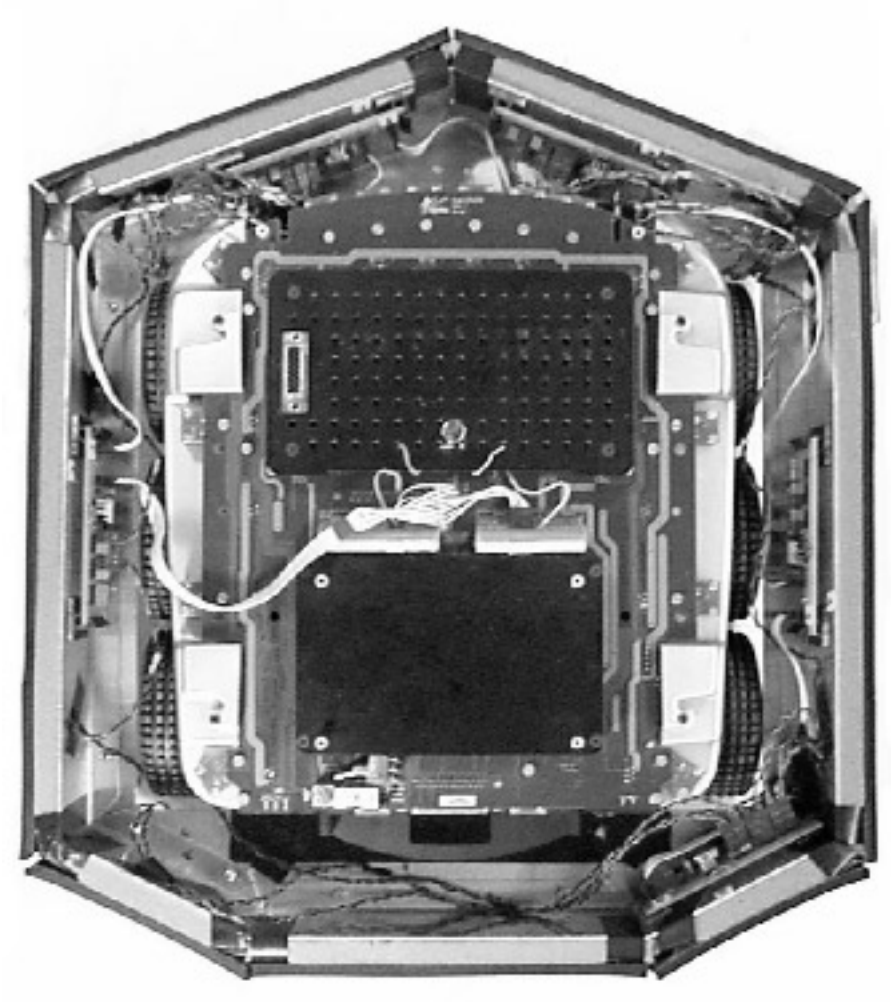

Figure A.1: Top view of the Koala robot with five sensor modules around it. With the modules attached, the robot is 46 cm long and 41 cm wide.

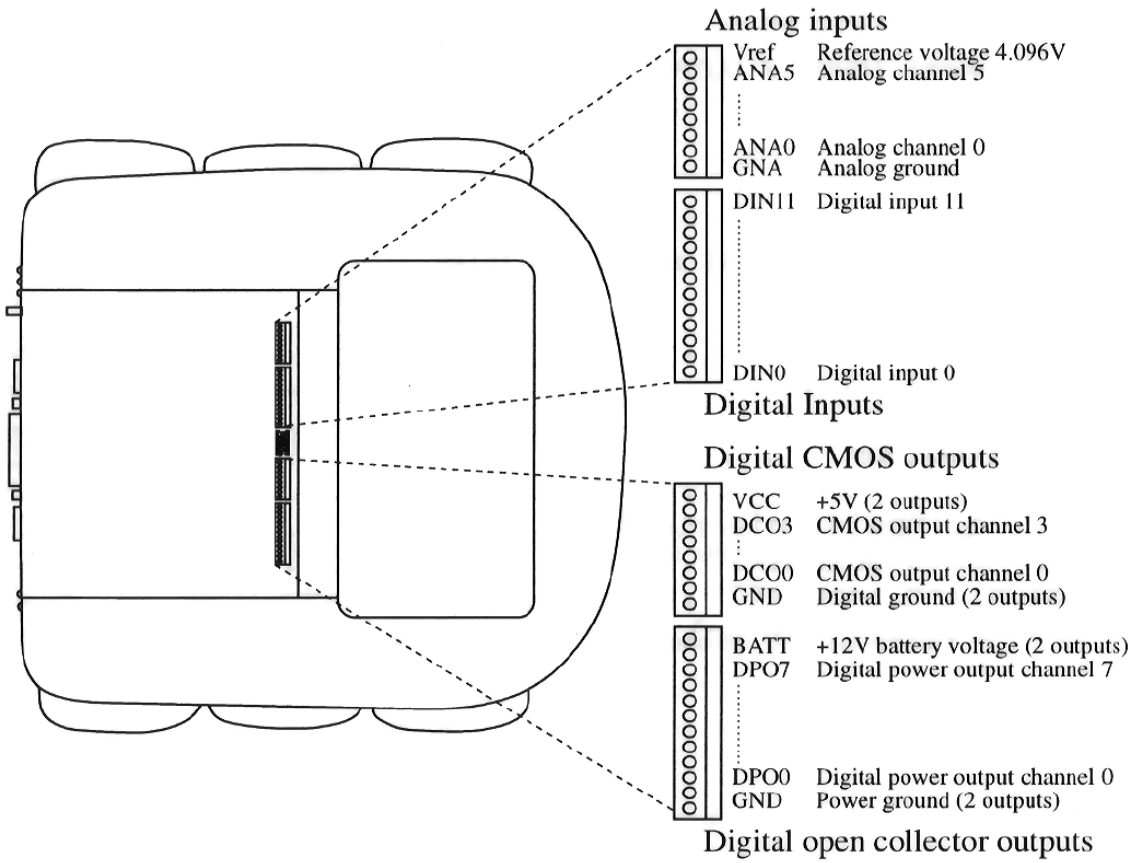

Figure A.2: The input/output connections of Koala, used for connecting the ribbon cable coming from modules. Picture taken from [9].

Appendix B

Electronic Circuit For The Sensors

This appendix contains information in making the PCB. Circuit diagram, PCB layout, and parts list are presented.

B.1 Connections

A ribbon cable connects all the modules to the Koala robot. Each module has different connections for PVDF sensors and because of that different numbers of resistors are used. The circuit diagram shown in Fig B.1 represents the circuit used for the PCB which was common for all the modules. However each module was specialised differently. The connections of PVDF to modules are presented in Table B.1,B.2 and B.3.

In these tables, JMP means a jumper. PVDF 2 means the PVDF film No. 2, in the layout. The numbers are illustrated for each configuration in Fig 3.5. The number of resistors represent how many resistors should be used in parallel with the films. If there are more positions available, the remaining positions should have a link inserted into them. Polarity of PVDF films are quite important when connecting them to the PCB. For CON2A,B,C the

Connector	Connections		
	1,2	3,4	Resistors
CON2A	PVDF 1	PVDF 4	3
CON2B	PVDF 2	JMP	1
CON2C	PVDF 3	JMP	1

Table B.1: PVDF connections for Front Config 1.

Connector	Connections		
	1,2	3,4	Resistors
CON2A	PVDF 1	JMP	2
CON2B	PVDF 2	JMP	2
CON2C	JMP	JMP	0

Table B.2: PVDF connections for Front Config 2 / Side. For CON2C there is no need to put links in place of resistors, since two jumpers are used anyway.

Connector	Connections		
	1,2	3,4	Resistors
CON2A	PVDF 1	PVDF 2	2
CON2B	PVDF 3	PVDF 4	2
CON2C	PVDF 5	PVDF 6	2

Table B.3: PVDF connections for Back.

Part Number	Component	Comments
R1-R9	10 MΩ	0.25W 5%
R10	560 KΩ	0.25W 5%
C1	100nF	Ceramic
C2-C5	100pF	Ceramic
D1-D3	1N4148	
J1-J10	Pinheaders	
U1	PIC12C671	Microchip
CON1A,CON1B	7 Pin Connectors	

Table B.4: The parts list for the PCB.

negative line of PVDF should be connected to PIN 2 or PIN 4[1].

In Fig B.2; J1,J2,J4,J5,J8 and J9 are ON if pin 1 and pin 2 are connected with a jumper, and they are OFF if pin 2 and pin 3 are connected.

[1]The red wire on PVDF is the negative line, it is manufactured this way!

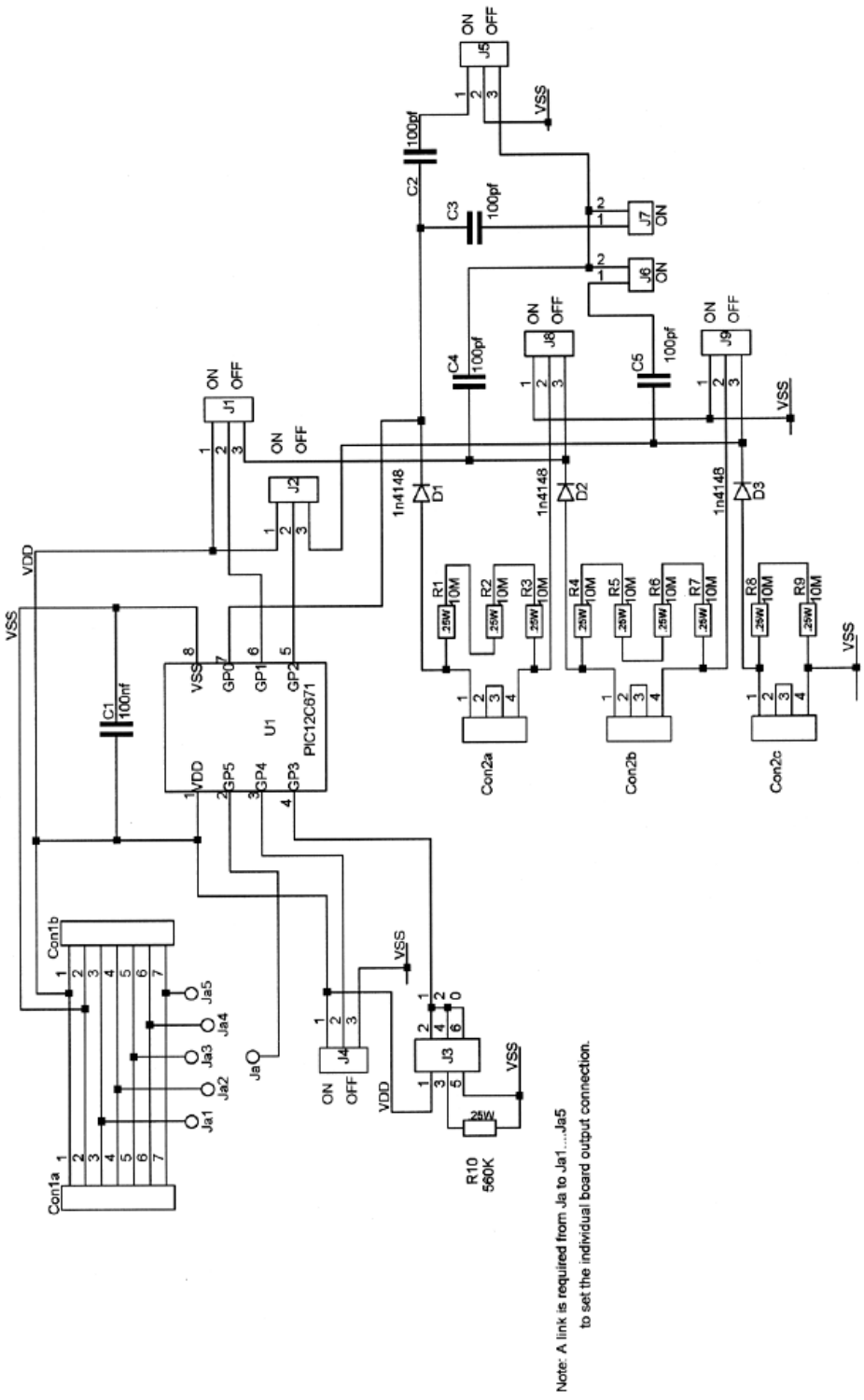

Figure B.1: Circuit diagram of the modules.

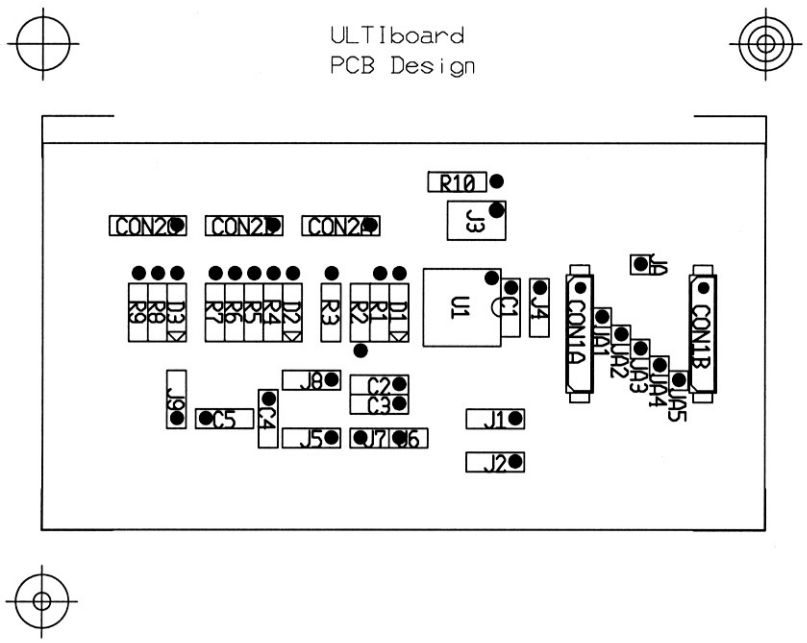

Figure B.2: Layout of PCB. It is single sided and dimensions are 10 cm × 6 cm. The dark dot represents pin 1 for each component.

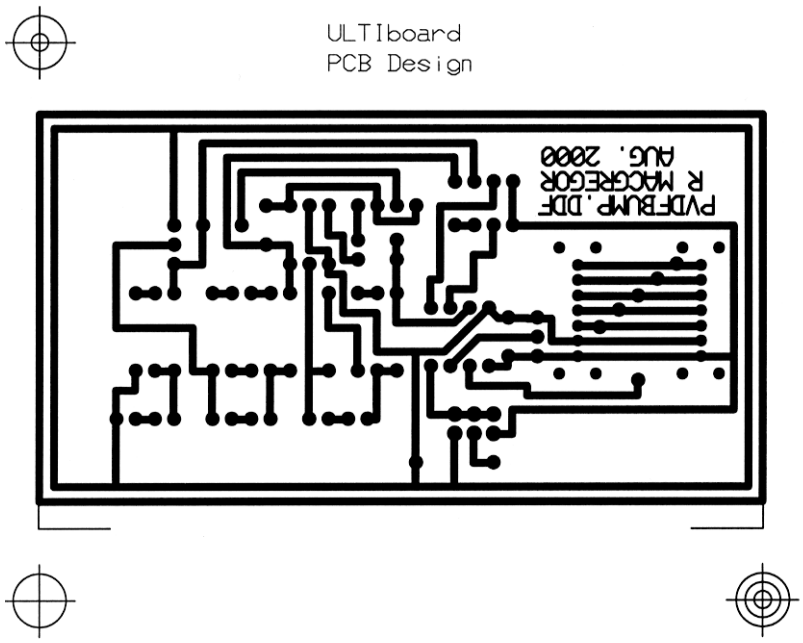

Figure B.3: Copper tracks of the PCB.

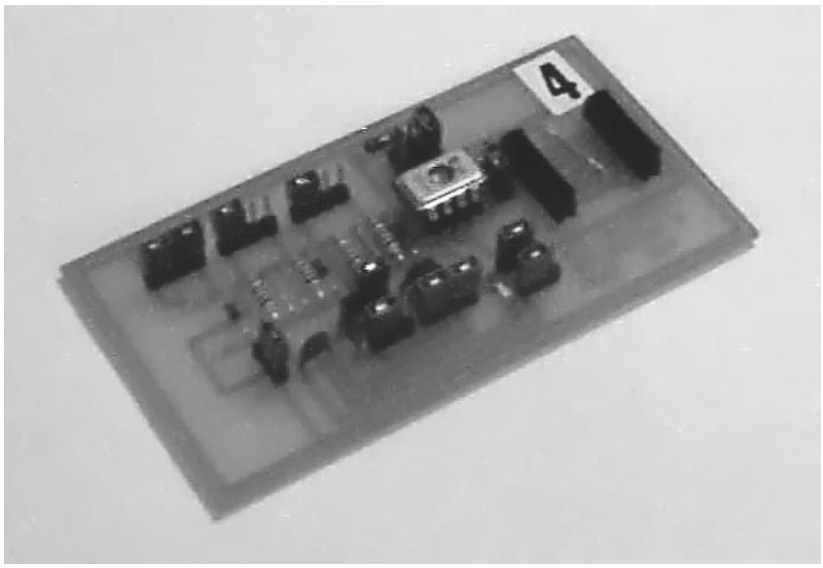

Figure B.4: Picture of the PCB used for every module.